CoNLL 2019

The 23rd Conference on Computational Natural Language Learning

Proceedings of the Conference

November 3–4, 2019
Hong Kong, China

Sponsors



©2019 The Association for Computational Linguistics

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-950737-72-7

Introduction

The 2019 Conference on Computational Natural Language Learning (CoNLL) is the 23rd in the series of annual meetings organized by SIGNLL, the ACL special interest group on natural language learning. CoNLL 2019 will be held on November 3–4, 2019, and is co-located with the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP) in Hong Kong.

CoNLL 2019 followed the tradition of previous CoNLL conferences in inviting only long papers, in order to accommodate papers with experimental material and detailed analysis. The final, camera-ready submissions were allowed a maximum of nine content pages plus unlimited pages of references and supplementary material.

CoNLL 2019 received a record number of 485 submissions in total, out of which 97 papers were chosen to appear in the conference program (after desk-rejections and a few papers withdrawn by the authors during the review period), with an overall acceptance rate of 22%. 27 were selected for oral presentation, and the remaining 70 for poster presentation. All 97 papers appear as long papers here in the conference proceedings.

CoNLL 2019 features two invited speakers, Christopher Manning (Stanford University) and Gabriella Vigliocco (University College London). As in recent years, it also features one shared task: Cross-Framework Meaning Representation Parsing. Papers accepted for the shared tasks are published in companion volumes of the CoNLL 2019 proceedings.

We would like to thank all the authors who submitted their work to CoNLL 2019, and the program committee for helping us select the best papers out of many high-quality submissions. We are grateful to the many program committee members who did a thorough job reviewing our submissions. Due to the growing size of of the conference, we also had area chairs, for the second time, supporting the CoNLL organization. We were fortunate to have 24 excellent areas chairs who assisted us greatly in selecting the best program:

Jason Baldridge, Google AI Language, USA;
Laurent Besacier, Université Grenoble Alpes, France;
Chris Biemann, Universität Hamburg, Germany;
Asli Celikyilmaz, Microsoft Research, USA;
Snigdha Chaturvedi, UCSC, USA;
Grzegorz Chrupala, Tilburg University, The Netherlands;
Mathieu Constant, Université de Lorraine, France;
Ryan Cotterell, University of Cambridge, UK;
Dipanjan Das, Google AI Language, USA;
Greg Durrett, UT Austin, USA;
Manaal Faruqui, Google Assistant, USA;
Michel Galley, Microsoft Research, USA;
Manuel Montes y Gómez, INAOE, Mexico;
Dilek Hakkani-Tur, Amazon Alexa AI, USA;
Mohit Iyyer, UMass Amherst, USA;
Yangfeng Ji, University of Virginia, USA;
Preethi Jyothi, IIT Bombay, India;
Douwe Kiela, Facebook Research, USA;
Graham Neubig, CMU, USA;
Horacio Saggion, Universitat Pompeu Fabra, Spain;
Avirup Sil, IBM Research AI, USA;
Amanda Stent, Bloomberg Research, USA;

Mark Stevenson, University of Sheffield, UK;
Andreas Vlachos, University of Cambridge, UK.

We are immensely thankful to Julia Hockenmaier and to the members of the SIGNLL board for their valuable advice and assistance in putting together this year's program. We also thank Pieter Fivez and Marcely Zanon Boito for maintaining the CoNLL 2019 website, and Sebastian Ruder and Miikka Silfverberg for preparing the proceedings for the main conference. We would like to thank our hard working assistants Darryl Hannan, Ramakanth Pasunuru and Reyhaneh Hashempour for their support with data checking and publicity. Our heartfelt gratitude also goes to Rodrigo Wilkens for system administration and general START management.

Our thanks to the program co-chairs of CoNLL 2018, Anna Korhonen and Ivan Titov, who provided us with excellent advice and help; to Vera Demberg, Naoaki Okazaki, Priscilla Rasmussen and the EMNLP 2019 Organization Committee for their helpful advice on issues involving the conference venue and local organization.

We would also like to thank the following reviewers who were nominated for commendation: Peter Anderson; Awais Athar; Niranjan Balasubramanian; Joost Bastings; Lisa Beinborn; Robert Berwick; Xavier Carreras; Elizabeth Clark; Pablo Duboue; Asif Ekbal; Zhe Gan; Dan Garrette; Sebastian Gehrmann; Kevin Gimpel; Carlos Gomez-Rodriguez; William L. Hamilton; David Harwath; Jack Hessel; Jonathan K. Kummerfeld; Miryam de Lhoneux; Nelson F. Liu; Ryan McDonald; Einat Minkov; Preslav Nakov; Jason Naradowsky; Khanh Nguyen; Vlad Niculae; Brendan O'Connor; Niki Parmar; Rebecca J. Passonneau; Iria del Rio Gayo; Kenji Sagae; Marten van Schijndel; Kevin Small; Kristina Striegnitz; James Thorne; Diyi Yang.

Finally, our gratitude goes to our sponsors, Facebook and Google, for supporting the conference financially.

We hope you enjoy the conference!

Aline Villavicencio and Mohit Bansal
CoNLL 2019 conference co-chairs

Conference Chairs:

Mohit Bansal, University of North Carolina at Chapel Hill, USA
Aline Villavicencio, University of Sheffield, UK and Federal University of Rio Grande do Sul, Brazil

Invited speakers:

Christopher Manning, Stanford University, USA
Gabriella Vigliocco, University College London, UK

Area Chairs:

Jason Baldridge, Google AI Language, USA
Laurent Besacier, Université Grenoble Alpes, France
Chris Biemann, Universität Hamburg, Germany
Asli Celikyilmaz, Microsoft Research, USA
Snigdha Chaturvedi, UCSC, USA
Grzegorz Chrupala, Tilburg University, The Netherlands
Mathieu Constant, Université de Lorraine, France
Ryan Cotterell, University of Cambridge, UK
Dipanjan Das, Google AI Language, USA
Greg Durrett, UT Austin, USA
Manaal Faruqui, Google Assistant, USA
Michel Galley, Microsoft Research, USA
Manuel Montes y Gómez, INAOE, Mexico
Dilek Hakkani-Tur, Amazon Alexa AI, USA
Mohit Iyyer, UMass Amherst, USA
Yangfeng Ji, University of Virginia, USA
Preethi Jyothi, IIT Bombay, India
Douwe Kiela, Facebook Research, USA
Graham Neubig, CMU, USA
Horacio Saggion, Universitat Pompeu Fabra, Spain
Avirup Sil, IBM Research AI, USA
Amanda Stent, Bloomberg Research, USA
Mark Stevenson, University of Sheffield, UK
Andreas Vlachos, University of Cambridge, UK

Publication Chairs:

Sebastian Ruder, National University of Ireland and Aylien Ltd. Dublin, Ireland
Miikka Silfverberg, University of Helsinki, Finland

Administration Chair:

Rodrigo Wilkens, University of Strasbourg, France

Supervision Chairs:

Darryl Hannan, University of North Carolina at Chapel Hill, USA
Reyhaneh Hashempour, University of Essex, UK

Publicity/Sponsorship Chair:

Ramakanth Pasunuru, University of North Carolina at Chapel Hill, USA

Website Chairs:

Marcely Zanon Boito, Université Grenoble Alpes, France
Pieter Fivez, University of Antwerp, Belgium

Program Committee:

Omri Abend, Ahmed AbuRa'ed, Pablo Accuosto, Heike Adel, Rodrigo Agerri, Eljko Agi, Aishwarya Agrawal, Roee Aharoni, Alan Akbik, Nader Akoury, Chris Alberti, Amal Alharbi, Afra Alishahi, Peter Anderson, Gabor Angeli, Saba Anwar, Marianna Apidianaki, Yuki Arase, Awais Athar, Fan Bai, Simon Baker, Niranjan Balasubramanian, Timothy Baldwin, Miguel Ballesteros, Colin Bannard, Francesco Barbieri, Leslie Barrett, Alberto Barrn-Cedeo, Fabian Barteld, Roberto Basili, Joost Bastings, David Batista, Timo Baumann, Barend Beekhuizen, Lisa Beinborn, Nria Bel, Jonathan Berant, Robert Berwick, Archna Bhatia, Pushpak Bhattacharyya, Lidong Bing, Philippe Blache, Eduardo Blanco, Bernd Bohnet, Danushka Bollegala, Kalina Bontcheva, Stefan Bott, Samuel R. Bowman, Faeze Brahman, Antnio Branco, Chlo Braud, lex Bravo, Chris Brockett, Elia Bruni, Harry Bunt, Davide Buscaldi, Jan Buys, Jose Camacho-Collados, Ricardo Campos, Cristian Cardellino, Xavier Carreras, Helena Caseli, Giovanni Cassani, Thiago Castro Ferreira, Asli Celikyilmaz, Daniel Cer, Muthu Kumar Chandrasekaran, Ming-Wei Chang, Yun-Nung Chen, Boxing Chen, Xinchi Chen, Hanjie Chen, Emmanuele Chersoni, Niyati Chhaya, Monojit Choudhury, George Chrysostomou, Volkan Cirik, Alexander Clark, Stephen Clark, Elizabeth Clark, Trevor Cohn, Guillem Collell, Danish Contractor, Paul Cook, Caio Corro, Marta R. Costajuss, Francisco M Couto, Raj Dabre, Walter Daelemans, Forrest Davis, Miryam de Lhoneux, Iria del Ro Gayo, Vera Demberg, Thomas Demeester, Nina Dethlefs, Daniel Deutsch, Jacob Devlin, Maria Pia di Buono, Shuoyang Ding, Simon Dobnik, Jesse Dodge, Lucia Donatelli, Li Dong, Zi-Yi Dou, Gabriel Doyle, Maximillian Droog-Hayes, Xinya Du, Pablo Duboue, Kevin Duh, Jonathan Dunn, Nadir Durrani, Richard Eckart de Castilho, Thomas Efer, Yo Ehara, Asif Ekbal, Ahmed El Kholy, Desmond Elliott, Micha Elsner, Chris Emmery, Erkut Erdem, Aykut Erdem, Akiko Eriguchi, Hugo Jair Escalante, Luis Espinosa Anke, Richard Evans, Benjamin Fagard, Stefano Faralli, Maryam Fazel-Zarandi, Christian Federmann, Yansong Feng, Raquel Fernndez, Orhan Firat, Andrea K. Fischer, Jeffrey Flanigan, Radu Florian, George Foster, Stella Frank, Diego Frassinelli, Adam Funk, Zhe Gan, Balaji Ganesan, Xiang Gao, Jianfeng Gao, Marcos Garcia, Dan Garrette, Sebastian Gehrmann, Lieke Gelderloos, Kim Gerdes, Mehdi Ghanimifard, Dafydd Gibbon, Daniel Gildea, Kevin Gimpel, Michael Glass, Goran Glava, Carlos Gmez-Rodrguez, Sharon Goldwater, Teresa Gonalves, Kartik Goyal, Pawan Goyal, Yvette Graham, Erin Grant, Mark Greenwood, Alvin Grissom II, Dagmar Gromann, Chulaka Gunasekara, Han Guo, Jiang Guo, Ankush Gupta, Iryna Gurevych, Gholamreza Haffari, ali hakimi parizi, William L. Hamilton, Benjamin Han, Darryl Hannan, David Harwath, Sadid A. Hasan, Mohammed Hasanuzzaman, Hua He, Luheng He, Drahomira Herrmannova, Jack Hessel, Vu Cong Duy Hoang, Eric Holgate, Ari Holtzman, Mark Hopkins, Renfen Hu, Xinyu Hua, Lifu Huang, Marco Idiart, Ozan Irsoy, Srinivasan Iyer, Cassandra L. Jacobs, Vihan Jain, Abhik Jana, Sharmistha Jat, Sujay Kumar Jauhar, Zhanming Jie, Anders Johannsen, alexander johansen, Aditya Joshi, Mandar Joshi, Jaap Kamps, Katharina Kann,

Diptesh Kanojia, Divyansh Kaushik, Daisuke Kawahara, Fabio Kepler, Daniel Khashabi, Seokhwan Kim, Yoon Kim, Milton King, Roman Klinger, Petr Knoth, Thomas Kober, Philipp Koehn, Rob Koeling, Rik Koncel-Kedziorski, Ioannis Konstas, Parisa Kordjamshidi, Yannis Korkontzelos, Leila Kosseim, Sachin Kumar, Jonathan K. Kummerfeld, Gourab Kundu, Tom Kwiatkowski, John P. Lalor, Ni Lao, Gabriella Lapesa, Alberto Lavelli, Phong Le, Yoong Keok Lee, Jason Lee, Jochen L. Leidner, Sarah Ita Levitan, Martha Lewis, Yanran Li, Jerry Li, Junyi Jessy Li, Marina Litvak, Nelson F. Liu, Yang Liu, Zhengzhong Liu, Elena Lloret, Chi-kiu Lo, Oier Lopez de Lacalle, David E. Losada, Adrin Pastor Lpez Monroy, Wei Lu, Michal Lukasik, Wencan Luo, Pranava Madhyastha, Giorgio Magri, Diego Marcheggiani, Stella Markantonatou, Katja Markert, David Martins de Matos, Yevgen Matusevych, Diana McCarthy, Arya D. McCarthy, David McClosky, R. Thomas Mccoy, Ryan McDonald, Stephen McGregor, Mohsen Mesgar, Sebastian J. Mielke, Einat Minkov, Dipendra Misra, Jeff Mitchell, Daichi Mochihashi, Marie-Francine Moens, Manuel Montes, Seungwhan Moon, Roser Morante, Alessandro Moschitti, Animesh Mukherjee, Smaranda Muresan, Kenton Murray, Preslav Nakov, Jason Naradowsky, Karthik Narasimhan, Shashi Narayan, Khanh Nguyen, Massimo Nicosia, Vlad Niculae, Jan Niehues, Andreas Niekler, Vassilina Nikoulina, Sergiu Nisioi, Tong Niu, Xing Niu, Brendan O'Connor, Kemal Oflazer, Constantin Orasan, Camilo Ortiz, Jessica Ouyang, Inkit Padhi, Aishwarya Padmakumar, Muntsa Padr, Alexander Panchenko, Alexandros Papangelis, Thiago Pardo, Ankur Parikh, Niki Parmar, Rebecca J. Passonneau, Ramakanth Pasunuru, Panupong Pasupat, Roma Patel, Amandalynne Paullada, Lisa Pearl, Anselmo Peas, Hao Peng, Nanyun Peng, Ethan Perez, Sandro Pezzelle, Janet Pierrehumbert, Mohammad Taher Pilehvar, Yuval Pinter, Lidia Pivovarova, Thierry Poibeau, Maja Popovi, Matt Post, Christopher Potts, Bruno Pouliquen, Vahed Qazvinian, Paulo Quaresma, Ella Rabinovich, Daniele P. Radicioni, Preethi Raghavan, Afshin Rahimi, Taraka Rama, Rohan Ramanath, Carlos Ramisch, Sudha Rao, Ari Rappoport, Mohammad Sadegh Rasooli, Sagnik Ray Choudhury, Andreas Rckl, Marek Rei, Roi Reichart, Steffen Remus, Xiang Ren, Horacio Rodriguez, Laurent Romary, Francesco Ronzano, Aiala Ros, Paolo Rosso, Michael Roth, Salim Roukos, Tania Roy, Subhro Roy, Alla Rozovskaya, Mrinmaya Sachan, Devendra Sachan, Mehrnoosh Sadrzadeh, Kenji Sagae, Diarmuid Saghdha, Magnus Sahlgren, Hassan Sajjad, Keisuke Sakaguchi, Sakriani Sakti, Rajhans Samdani, Ivan Sanchez, Carolina Scarton, Natalie Schluter, Nathan Schneider, Steven Schockaert, Djam Seddah, Marco Silvio Giuseppe Senaldi, zge Sevgili, Amr Sharaf, Dinghan Shen, Wei Shi, Alexander Shvets, Carina Silberer, Miikka Silfverberg, Jonathan Simon, Kevin Small, Artem Sokolov, Lucia Specia, Vivek Srikumar, Shashank Srivastava, Efstathios Stamatatos, Milo Stanojevi, Gabriel Stanovsky, Egon Stemle, Suzanne Stevenson, Karl Stratos, Kristina Striegnitz, Pei-Hao Su, Shivashankar Subramanian, Alane Suhr, Anas Tack, Jiwei Tan, Christoph Teichmann, Ian Tenney, Jesse Thomason, James Thorne, Ran Tian, Amalia Todirascu, Gaurav Singh Tomar, Juan-Manuel Torres-Moreno, Harsh Trivedi, Gokhan Tur, Shyam Upadhyay, Tim Van de Cruys, Marten van Schijndel, Lucy Vanderwende, Shikhar Vashishth, Ramakrishna Vedantam, Yannick Versley, Supriya Vijay, David Vilar, Esau Villatoro-Tello, Marta Villegas, Tatiana Vodolazova, Tu Vu, Ivan Vuli, Xin Wang, Miaosen Wang, Leo Wanner, Taro Watanabe, Austin Waters, Noah Weber, Kellie Webster, Michael Wiegand, John Wieting, Gijs Wijnholds, Rodrigo Wilkens, Steven Wilson, Sam Wiseman, Vinicius Woloszyn, Dekai Wu, Kun Xu, Yang Xu, Yadollah Yaghoobzadeh, Mohamed Yahya, Rui Yan, Jie Yang, Diyi Yang, Weiwei Yang, Roman Yangarber, Ziyu Yao, Semih Yavuz, Seid Yimam, Wenpeng Yin, Zhou Yu, Licheng Yu, Franois Yvon, Roberto Zamparelli, Marcos Zampieri, Neil Zeghidour, Luke Zettlemoyer, Feifei Zhai, Yuan Zhang, Xingxing Zhang, Zhisong Zhang, Yizhe Zhang, Yue Zhang, Wei Zhao, Tiancheng Zhao, Kai Zhao, Chao Zhao, Steven Zimmerman, Heike Zinsmeister, Michael Zock, Chengqing Zong, Shi Zong, and Willem Zuidema.

Table of Contents

Invited Talk I: Ecological Language: A Multimodal Approach to the Study of Human Language Learning and Processing
Gabriella Vigliocco xxvi

Invited Talk II: Multi-Step Reasoning for Answering Complex Questions
Christopher Manning xxvii

Analysing Neural Language Models: Contextual Decomposition Reveals Default Reasoning in Number and Gender Assignment
Jaap Jumelet, Willem Zuidema and Dieuwke Hupkes 1

Deconstructing Supertagging into Multi-Task Sequence Prediction
Zhenqi Zhu and Anoop Sarkar 12

Multilingual Model Using Cross-Task Embedding Projection
Jin Sakuma and Naoki Yoshinaga 22

Investigating Cross-Lingual Alignment Methods for Contextualized Embeddings with Token-Level Evaluation
Qianchu Liu, Diana McCarthy, Ivan Vulić and Anna Korhonen 33

Large-Scale, Diverse, Paraphrastic Bitexts via Sampling and Clustering
J. Edward Hu, Abhinav Singh, Nils Holzenberger, Matt Post and Benjamin Van Durme 44

Large-Scale Representation Learning from Visually Grounded Untranscribed Speech
Gabriel Ilharco, Yuan Zhang and Jason Baldridge 55

Using Priming to Uncover the Organization of Syntactic Representations in Neural Language Models
Grusha Prasad, Marten van Schijndel and Tal Linzen 66

Say Anything: Automatic Semantic Infelicity Detection in L2 English Indefinite Pronouns
Ella Rabinovich, Julia Watson, Barend Beekhuizen and Suzanne Stevenson 77

Compositional Generalization in Image Captioning
Mitja Nikolaus, Mostafa Abdou, Matthew Lamm, Rahul Aralikatte and Desmond Elliott 87

Representing Movie Characters in Dialogues
Mahmoud Azab, Noriyuki Kojima, Jia Deng and Rada Mihalcea 99

Cross-Lingual Word Embeddings and the Structure of the Human Bilingual Lexicon
Paola Merlo and Maria Andueza Rodriguez 110

Federated Learning of N-Gram Language Models
Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays and Michael Riley 121

Learning Conceptual Spaces with Disentangled Facets
Rana Alshaikh, Zied Bouraoui and Steven Schockaert 131

Weird Inflects but OK: Making Sense of Morphological Generation Errors
Kyle Gorman, Arya D. McCarthy, Ryan Cotterell, Ekaterina Vylomova, Miikka Silfverberg and Magdalena Markowska 140

Learning to Represent Bilingual Dictionaries
Muhao Chen, Yingtao Tian, Haochen Chen, Kai-Wei Chang, Steven Skiena and Carlo Zaniolo 152

Improving Natural Language Understanding by Reverse Mapping Bytepair Encoding
Chaodong Tong, Huailiang Peng, Qiong Dai, Lei Jiang and Jianghua Huang 163

Made for Each Other: Broad-Coverage Semantic Structures Meet Preposition Supersenses
Jakob Prange, Nathan Schneider and Omri Abend 174

Generating Timelines by Modeling Semantic Change
Guy D. Rosin and Kira Radinsky 186

Diversify Your Datasets: Analyzing Generalization via Controlled Variance in Adversarial Datasets
Ohad Rozen, Vered Shwartz, Roee Aharoni and Ido Dagan 196

Fully Unsupervised Crosslingual Semantic Textual Similarity Metric Based on BERT for Identifying Parallel Data
Chi-kiu Lo and Michel Simard 206

On the Importance of Subword Information for Morphological Tasks in Truly Low-Resource Languages
Yi Zhu, Benjamin Heinzerling, Ivan Vulić, Michael Strube, Roi Reichart and Anna Korhonen . 216

Comparing Top-Down and Bottom-Up Neural Generative Dependency Models
Austin Matthews, Graham Neubig and Chris Dyer 227

Representation Learning and Dynamic Programming for Arc-Hybrid Parsing
Joseph Le Roux, Antoine Rozenknop and Mathieu Lacroix 238

Policy Preference Detection in Parliamentary Debate Motions
Gavin Abercrombie, Federico Nanni, Riza Batista-Navarro and Simone Paolo Ponzetto 249

Improving Neural Machine Translation by Achieving Knowledge Transfer with Sentence Alignment Learning
Xuewen Shi, Heyan Huang, Wenguan Wang, Ping Jian and Yi-Kun Tang 260

Code-Switched Language Models Using Neural Based Synthetic Data from Parallel Sentences
Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu and Pascale Fung 271

Unsupervised Neural Machine Translation with Future Rewarding
Xiangpeng Wei, Yue Hu, Luxi Xing and Li Gao 281

Automatically Extracting Challenge Sets for Non-Local Phenomena in Neural Machine Translation
Leshem Choshen and Omri Abend 291

Low-Resource Parsing with Crosslingual Contextualized Representations
Phoebe Mulcaire, Jungo Kasai and Noah A. Smith 304

Improving Pre-Trained Multilingual Model with Vocabulary Expansion
Hai Wang, Dian Yu, Kai Sun, Jianshu Chen and Dong Yu 316

On the Relation between Position Information and Sentence Length in Neural Machine Translation
Masato Neishi and Naoki Yoshinaga 328

Word Recognition, Competition, and Activation in a Model of Visually Grounded Speech
William N. Havard, Jean-Pierre Chevrot and Laurent Besacier 339

EQUATE: A Benchmark Evaluation Framework for Quantitative Reasoning in Natural Language Inference
Abhilasha Ravichander, Aakanksha Naik, Carolyn Rose and Eduard Hovy 349

Linguistic Analysis Improves Neural Metaphor Detection
Kevin Stowe, Sarah Moeller, Laura Michaelis and Martha Palmer 362

Cross-Lingual Dependency Parsing with Unlabeled Auxiliary Languages
Wasi Uddin Ahmad, Zhisong Zhang, Xuezhe Ma, Kai-Wei Chang and Nanyun Peng 372

A Dual-Attention Hierarchical Recurrent Neural Network for Dialogue Act Classification
Ruizhe Li, Chenghua Lin, Matthew Collinson, Xiao Li and Guanyi Chen 383

Mimic and Rephrase: Reflective Listening in Open-Ended Dialogue
Justin Dieter, Tian Wang, Arun Tejasvi Chaganty, Gabor Angeli and Angel X. Chang 393

Automated Pyramid Summarization Evaluation
Yanjun Gao, Chen Sun and Rebecca J. Passonneau 404

A Case Study on Combining ASR and Visual Features for Generating Instructional Video Captions
Jack Hessel, Bo Pang, Zhenhai Zhu and Radu Soricut 419

Leveraging Past References for Robust Language Grounding
Subhro Roy, Michael Noseworthy, Rohan Paul, Daehyung Park and Nicholas Roy 430

Procedural Reasoning Networks for Understanding Multimodal Procedures
Mustafa Sercan Amac, Semih Yagcioglu, Aykut Erdem and Erkut Erdem 441

On the Limits of Learning to Actively Learn Semantic Representations
Omri Koshorek, Gabriel Stanovsky, Yichu Zhou, Vivek Srikumar and Jonathan Berant 452

How Does Grammatical Gender Affect Noun Representations in Gender-Marking Languages?
Hila Gonen, Yova Kementchedjhieva and Yoav Goldberg 463

Active Learning via Membership Query Synthesis for Semi-Supervised Sentence Classification
Raphael Schumann and Ines Rehbein 472

A General-Purpose Algorithm for Constrained Sequential Inference
Daniel Deutsch, Shyam Upadhyay and Dan Roth 482

A Richly Annotated Corpus for Different Tasks in Automated Fact-Checking
Andreas Hanselowski, Christian Stab, Claudia Schulz, Zile Li and Iryna Gurevych 493

Detecting Frames in News Headlines and Its Application to Analyzing News Framing Trends Surrounding U.S. Gun Violence
Siyi Liu, Lei Guo, Kate Mays, Margrit Betke and Derry Tanti Wijaya 504

Learning a Unified Named Entity Tagger from Multiple Partially Annotated Corpora for Efficient Adaptation
Xiao Huang, Li Dong, Elizabeth Boschee and Nanyun Peng 515

Learning Dense Representations for Entity Retrieval
Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie and Diego Garcia-Olano 528

CogniVal: A Framework for Cognitive Word Embedding Evaluation
Nora Hollenstein, Antonio de la Torre, Nicolas Langer and Ce Zhang 538

KnowSemLM: A Knowledge Infused Semantic Language Model
Haoruo Peng, Qiang Ning and Dan Roth 550

Neural Attentive Bag-of-Entities Model for Text Classification
Ikuya Yamada and Hiroyuki Shindo 563

Roll Call Vote Prediction with Knowledge Augmented Models
Pallavi Patil, Kriti Myer, Ronak Zala, Arpit Singh, Sheshera Mysore, Andrew McCallum, Adrian Benton and Amanda Stent 574

BeamSeg: A Joint Model for Multi-Document Segmentation and Topic Identification
Pedro Mota, Maxine Eskenazi and Luísa Coheur 582

MrMep: Joint Extraction of Multiple Relations and Multiple Entity Pairs Based on Triplet Attention
Jiayu Chen, Caixia Yuan, Xiaojie Wang and Ziwei Bai 593

Effective Attention Modeling for Neural Relation Extraction
Tapas Nayak and Hwee Tou Ng 603

Exploiting the Entity Type Sequence to Benefit Event Detection
Yuze Ji, Youfang Lin, Jianwei Gao and Huaiyu Wan 613

Named Entity Recognition - Is There a Glass Ceiling?
Tomasz Stanislawek, Anna Wróblewska, Alicja Wójcicka, Daniel Ziembicki and Przemyslaw Biecek 624

Low-Rank Approximations of Second-Order Document Representations
Jarkko Lagus, Janne Sinkkonen and Arto Klami 634

Named Entity Recognition with Partially Annotated Training Data
Stephen Mayhew, Snigdha Chaturvedi, Chen-Tse Tsai and Dan Roth 645

Contextualized Cross-Lingual Event Trigger Extraction with Minimal Resources
Meryem M'hamdi, Marjorie Freedman and Jonathan May 656

Deep Structured Neural Network for Event Temporal Relation Extraction
Rujun Han, I-Hung Hsu, Mu Yang, Aram Galstyan, Ralph Weischedel and Nanyun Peng 666

Investigating Entity Knowledge in BERT with Simple Neural End-To-End Entity Linking
Samuel Broscheit 677

Unsupervised Adversarial Domain Adaptation for Implicit Discourse Relation Classification
Hsin-Ping Huang and Junyi Jessy Li 686

Evidence Sentence Extraction for Machine Reading Comprehension
Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, Dong Yu, David McAllester and Dan Roth 696

SimVecs: Similarity-Based Vectors for Utterance Representation in Conversational AI Systems
Ashraf Mahgoub, Youssef Shahin, Riham Mansour and Saurabh Bagchi 708

Incorporating Interlocutor-Aware Context into Response Generation on Multi-Party Chatbots
Cao Liu, Kang Liu, Shizhu He, Zaiqing Nie and Jun Zhao 718

Memory Graph Networks for Explainable Memory-grounded Question Answering
Seungwhan Moon, Pararth Shah, Anuj Kumar and Rajen Subba 728

TripleNet: Triple Attention Network for Multi-Turn Response Selection in Retrieval-Based Chatbots
Wentao Ma, Yiming Cui, Nan Shao, Su He, Wei-Nan Zhang, Ting Liu, Shijin Wang and Guoping Hu 737

Relation Module for Non-Answerable Predictions on Reading Comprehension
Kevin Huang, Yun Tang, Jing Huang, Xiaodong He and Bowen Zhou 747

Slot Tagging for Task Oriented Spoken Language Understanding in Human-to-Human Conversation Scenarios
Kunho Kim, Rahul Jha, Kyle Williams, Alex Marin and Imed Zitouni 757

Window-Based Neural Tagging for Shallow Discourse Argument Labeling
René Knaebel, Manfred Stede and Sebastian Stober 768

TILM: Neural Language Models with Evolving Topical Influence
Shubhra Kanti Karmaker Santu, Kalyan Veeramachaneni and Chengxiang Zhai 778

Pretraining-Based Natural Language Generation for Text Summarization
Haoyu Zhang, Jingjing Cai, Jianjun Xu and Ji Wang 789

Goal-Embedded Dual Hierarchical Model for Task-Oriented Dialogue Generation
Yi-An Lai, Arshit Gupta and Yi Zhang 798

Putting the Horse before the Cart: A Generator-Evaluator Framework for Question Generation from Text
Vishwajeet Kumar, Ganesh Ramakrishnan and Yuan-Fang Li 812

In Conclusion Not Repetition: Comprehensive Abstractive Summarization with Diversified Attention Based on Determinantal Point Processes
Lei Li, Wei Liu, Marina Litvak, Natalia Vanetik and Zuying Huang 822

Generating Formality-Tuned Summaries Using Input-Dependent Rewards
Kushal Chawla, Balaji Vasan Srinivasan and Niyati Chhaya 833

Do Massively Pretrained Language Models Make Better Storytellers?
Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola and Christopher D. Manning 843

Self-Adaptive Scaling for Learnable Residual Structure
Fenglin Liu, Meng Gao, Yuanxin Liu and Kai Lei 862

BIOfid Dataset: Publishing a German Gold Standard for Named Entity Recognition in Historical Biodiversity Literature
Sajawel Ahmed, Manuel Stoeckel, Christine Driller, Adrian Pachzelt and Alexander Mehler . . 871

Slang Detection and Identification
Zhengqi Pei, Zhewei Sun and Yang Xu 881

Alleviating Sequence Information Loss with Data Overlapping and Prime Batch Sizes
Noémien Kocher, Christian Scuito, Lorenzo Tarantino, Alexandros Lazaridis, Andreas Fischer and Claudiu Musat 890

Global Autoregressive Models for Data-Efficient Sequence Learning
Tetiana Parshakova, Jean-Marc Andreoli and Marc Dymetman 900

Learning Analogy-Preserving Sentence Embeddings for Answer Selection
Aïssatou Diallo, Markus Zopf and Johannes Fürnkranz 910

A Simple and Effective Method for Injecting Word-Level Information into Character-Aware Neural Language Models
Yukun Feng, Hidetaka Kamigaito, Hiroya Takamura and Manabu Okumura 920

On Model Stability as a Function of Random Seed
Pranava Madhyastha and Rishabh Jain 929

Studying Generalisability across Abusive Language Detection Datasets
Steve Durairaj Swamy, Anupam Jamatia and Björn Gambäck 940

Reduce & Attribute: Two-Step Authorship Attribution for Large-Scale Problems
Michael Tschuggnall, Benjamin Murauer and Günther Specht 951

Variational Semi-Supervised Aspect-Term Sentiment Analysis via Transformer
Xingyi Cheng, Weidi Xu, Taifeng Wang, Wei Chu, Weipeng Huang, Kunlong Chen and Junfeng Hu 961

Learning to Detect Opinion Snippet for Aspect-Based Sentiment Analysis
Mengting Hu, Shiwan Zhao, Honglei Guo, Renhong Cheng and Zhong Su 970

Multi-Level Sentiment Analysis of PolEmo 2.0: Extended Corpus of Multi-Domain Consumer Reviews
Jan Kocoń, Piotr Miłkowski and Monika Zaśko-Zielińska 980

A Personalized Sentiment Model with Textual and Contextual Information
Siwen Guo, Sviatlana Höhn and Christoph Schommer 992

Cluster-Gated Convolutional Neural Network for Short Text Classification
Haidong Zhang, Wancheng Ni, Meijing Zhao and Ziqi Lin 1002

Coherence-Based Modeling of Clinical Concepts Inferred from Heterogeneous Clinical Notes for ICU Patient Risk Stratification
Tushaar Gangavarapu, Gokul S Krishnan and Sowmya Kamath 1012

Predicting the Role of Political Trolls in Social Media
Atanas Atanasov, Gianmarco De Francisci Morales and Preslav Nakov 1023

Towards a Unified End-to-End Approach for Fully Unsupervised Cross-Lingual Sentiment Analysis
Yanlin Feng and Xiaojun Wan 1035

Conference Program

Sunday, November 3, 2019

8:45–9:00 *Opening session*
Aline Villavicencio and Mohit Bansal

9:00–10:30 Session 1

9:00–9:15 *Analysing Neural Language Models: Contextual Decomposition Reveals Default Reasoning in Number and Gender Assignment*
Jaap Jumelet, Willem Zuidema and Dieuwke Hupkes

9:15–9:30 *Deconstructing Supertagging into Multi-Task Sequence Prediction*
Zhenqi Zhu and Anoop Sarkar

9:30–9:45 *Multilingual Model Using Cross-Task Embedding Projection*
Jin Sakuma and Naoki Yoshinaga

9:45–10:00 *Investigating Cross-Lingual Alignment Methods for Contextualized Embeddings with Token-Level Evaluation*
Qianchu Liu, Diana McCarthy, Ivan Vulić and Anna Korhonen

10:00–10:15 *Large-Scale, Diverse, Paraphrastic Bitexts via Sampling and Clustering*
J. Edward Hu, Abhinav Singh, Nils Holzenberger, Matt Post and Benjamin Van Durme

10:15–10:30 *Large-Scale Representation Learning from Visually Grounded Untranscribed Speech*
Gabriel Ilharco, Yuan Zhang and Jason Baldridge

Sunday, November 3, 2019 (continued)

10:30–11:00 Coffee Break

11:00–12:00 Invited Speaker

11:00–12:00 *Invited Talk: Ecological Language: a multimodal approach to the study of human language learning and processing*
Gabriella Vigliocco

12:00–12:30 Session 2

12:00–12:15 *Using Priming to Uncover the Organization of Syntactic Representations in Neural Language Models*
Grusha Prasad, Marten van Schijndel and Tal Linzen

12:15–12:30 *Say Anything: Automatic Semantic Infelicity Detection in L2 English Indefinite Pronouns*
Ella Rabinovich, Julia Watson, Barend Beekhuizen and Suzanne Stevenson

12:30–14:00 Lunch

14:00–15:30 CoNLL 2019 Shared Task: Cross-Framework Meaning Representation Parsing (MRP 2019)

15:30–16:00 Coffee Break

Sunday, November 3, 2019 (continued)

16:00–16:30 Session 3

16:00–16:15 *Compositional Generalization in Image Captioning*
Mitja Nikolaus, Mostafa Abdou, Matthew Lamm, Rahul Aralikatte and Desmond Elliott

16:15–16:30 *Representing Movie Characters in Dialogues*
Mahmoud Azab, Noriyuki Kojima, Jia Deng and Rada Mihalcea

16:30–18:00 Poster Session 1

16:30–18:00 *Cross-Lingual Word Embeddings and the Structure of the Human Bilingual Lexicon*
Paola Merlo and Maria Andueza Rodriguez

16:30–18:00 *Federated Learning of N-Gram Language Models*
Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong, Cyril Allauzen, Françoise Beaufays and Michael Riley

16:30–18:00 *Learning Conceptual Spaces with Disentangled Facets*
Rana Alshaikh, Zied Bouraoui and Steven Schockaert

16:30–18:00 *Weird Inflects but OK: Making Sense of Morphological Generation Errors*
Kyle Gorman, Arya D. McCarthy, Ryan Cotterell, Ekaterina Vylomova, Miikka Silfverberg and Magdalena Markowska

16:30–18:00 *Learning to Represent Bilingual Dictionaries*
Muhao Chen, Yingtao Tian, Haochen Chen, Kai-Wei Chang, Steven Skiena and Carlo Zaniolo

16:30–18:00 *Improving Natural Language Understanding by Reverse Mapping Bytepair Encoding*
Chaodong Tong, Huailiang Peng, Qiong Dai, Lei Jiang and Jianghua Huang

16:30–18:00 *Made for Each Other: Broad-Coverage Semantic Structures Meet Preposition Supersenses*
Jakob Prange, Nathan Schneider and Omri Abend

16:30–18:00 *Generating Timelines by Modeling Semantic Change*
Guy D. Rosin and Kira Radinsky

Sunday, November 3, 2019 (continued)

16:30–18:00 *Diversify Your Datasets: Analyzing Generalization via Controlled Variance in Adversarial Datasets*
Ohad Rozen, Vered Shwartz, Roee Aharoni and Ido Dagan

16:30–18:00 *Fully Unsupervised Crosslingual Semantic Textual Similarity Metric Based on BERT for Identifying Parallel Data*
Chi-kiu Lo and Michel Simard

16:30–18:00 *On the Importance of Subword Information for Morphological Tasks in Truly Low-Resource Languages*
Yi Zhu, Benjamin Heinzerling, Ivan Vulić, Michael Strube, Roi Reichart and Anna Korhonen

16:30–18:00 *Comparing Top-Down and Bottom-Up Neural Generative Dependency Models*
Austin Matthews, Graham Neubig and Chris Dyer

16:30–18:00 *Representation Learning and Dynamic Programming for Arc-Hybrid Parsing*
Joseph Le Roux, Antoine Rozenknop and Mathieu Lacroix

16:30–18:00 *Policy Preference Detection in Parliamentary Debate Motions*
Gavin Abercrombie, Federico Nanni, Riza Batista-Navarro and Simone Paolo Ponzetto

16:30–18:00 *Improving Neural Machine Translation by Achieving Knowledge Transfer with Sentence Alignment Learning*
Xuewen Shi, Heyan Huang, Wenguan Wang, Ping Jian and Yi-Kun Tang

16:30–18:00 *Code-Switched Language Models Using Neural Based Synthetic Data from Parallel Sentences*
Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu and Pascale Fung

16:30–18:00 *Unsupervised Neural Machine Translation with Future Rewarding*
Xiangpeng Wei, Yue Hu, Luxi Xing and Li Gao

16:30–18:00 *Automatically Extracting Challenge Sets for Non-Local Phenomena in Neural Machine Translation*
Leshem Choshen and Omri Abend

16:30–18:00 *Low-Resource Parsing with Crosslingual Contextualized Representations*
Phoebe Mulcaire, Jungo Kasai and Noah A. Smith

16:30–18:00 *Improving Pre-Trained Multilingual Model with Vocabulary Expansion*
Hai Wang, Dian Yu, Kai Sun, Jianshu Chen and Dong Yu

Sunday, November 3, 2019 (continued)

16:30–18:00 *On the Relation between Position Information and Sentence Length in Neural Machine Translation*
Masato Neishi and Naoki Yoshinaga

16:30–18:00 *Word Recognition, Competition, and Activation in a Model of Visually Grounded Speech*
William N. Havard, Jean-Pierre Chevrot and Laurent Besacier

16:30–18:00 *EQUATE: A Benchmark Evaluation Framework for Quantitative Reasoning in Natural Language Inference*
Abhilasha Ravichander, Aakanksha Naik, Carolyn Rose and Eduard Hovy

16:30–18:00 *Linguistic Analysis Improves Neural Metaphor Detection*
Kevin Stowe, Sarah Moeller, Laura Michaelis and Martha Palmer

18:00–18:30 Reception

Monday, November 4, 2019

8:45–10:30 Session 4

8:45–9:00 *Cross-Lingual Dependency Parsing with Unlabeled Auxiliary Languages*
Wasi Uddin Ahmad, Zhisong Zhang, Xuezhe Ma, Kai-Wei Chang and Nanyun Peng

9:00–9:15 *A Dual-Attention Hierarchical Recurrent Neural Network for Dialogue Act Classification*
Ruizhe Li, Chenghua Lin, Matthew Collinson, Xiao Li and Guanyi Chen

9:15–9:30 *Mimic and Rephrase: Reflective Listening in Open-Ended Dialogue*
Justin Dieter, Tian Wang, Arun Tejasvi Chaganty, Gabor Angeli and Angel X. Chang

9:30–9:45 *Automated Pyramid Summarization Evaluation*
Yanjun Gao, Chen Sun and Rebecca J. Passonneau

9:45–10:00 *A Case Study on Combining ASR and Visual Features for Generating Instructional Video Captions*
Jack Hessel, Bo Pang, Zhenhai Zhu and Radu Soricut

10:00–10:15 *Leveraging Past References for Robust Language Grounding*
Subhro Roy, Michael Noseworthy, Rohan Paul, Daehyung Park and Nicholas Roy

Monday, November 4, 2019 (continued)

10:15–10:30 *Procedural Reasoning Networks for Understanding Multimodal Procedures*
Mustafa Sercan Amac, Semih Yagcioglu, Aykut Erdem and Erkut Erdem

10:30–11:00 Coffee Break

11:00–12:00 Invited Speaker

11:00–12:00 *Invited Talk: Multi-step reasoning for answering complex questions*
Chris Manning

12:00–12:30 Session 5

12:00–12:15 *On the Limits of Learning to Actively Learn Semantic Representations*
Omri Koshorek, Gabriel Stanovsky, Yichu Zhou, Vivek Srikumar and Jonathan Berant

12:15–12:30 *How Does Grammatical Gender Affect Noun Representations in Gender-Marking Languages?*
Hila Gonen, Yova Kementchedjhieva and Yoav Goldberg

12:30–14:00 Best Paper Awards and Community Business Meeting

14:00–15:30 Session 6

14:00–14:15 *Active Learning via Membership Query Synthesis for Semi-Supervised Sentence Classification*
Raphael Schumann and Ines Rehbein

14:15–14:30 *A General-Purpose Algorithm for Constrained Sequential Inference*
Daniel Deutsch, Shyam Upadhyay and Dan Roth

14:30–14:45 *A Richly Annotated Corpus for Different Tasks in Automated Fact-Checking*
Andreas Hanselowski, Christian Stab, Claudia Schulz, Zile Li and Iryna Gurevych

14:45–15:00 *Detecting Frames in News Headlines and Its Application to Analyzing News Framing Trends Surrounding U.S. Gun Violence*
Siyi Liu, Lei Guo, Kate Mays, Margrit Betke and Derry Tanti Wijaya

Monday, November 4, 2019 (continued)

15:00–15:15 *Learning a Unified Named Entity Tagger from Multiple Partially Annotated Corpora for Efficient Adaptation*
Xiao Huang, Li Dong, Elizabeth Boschee and Nanyun Peng

15:15–15:30 *Learning Dense Representations for Entity Retrieval*
Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie and Diego Garcia-Olano

15:30–16:00 Coffee Break

16:00–16:30 Session 7

16:00–16:15 *CogniVal: A Framework for Cognitive Word Embedding Evaluation*
Nora Hollenstein, Antonio de la Torre, Nicolas Langer and Ce Zhang

16:15–16:30 *KnowSemLM: A Knowledge Infused Semantic Language Model*
Haoruo Peng, Qiang Ning and Dan Roth

16:30–18:00 Poster Session 2

16:30–18:00 *Neural Attentive Bag-of-Entities Model for Text Classification*
Ikuya Yamada and Hiroyuki Shindo

16:30–18:00 *Roll Call Vote Prediction with Knowledge Augmented Models*
Pallavi Patil, Kriti Myer, Ronak Zala, Arpit Singh, Sheshera Mysore, Andrew McCallum, Adrian Benton and Amanda Stent

16:30–18:00 *BeamSeg: A Joint Model for Multi-Document Segmentation and Topic Identification*
Pedro Mota, Maxine Eskenazi and Luísa Coheur

16:30–18:00 *MrMep: Joint Extraction of Multiple Relations and Multiple Entity Pairs Based on Triplet Attention*
Jiayu Chen, Caixia Yuan, Xiaojie Wang and Ziwei Bai

16:30–18:00 *Effective Attention Modeling for Neural Relation Extraction*
Tapas Nayak and Hwee Tou Ng

Monday, November 4, 2019 (continued)

16:30–18:00 *Exploiting the Entity Type Sequence to Benefit Event Detection*
Yuze Ji, Youfang Lin, Jianwei Gao and Huaiyu Wan

16:30–18:00 *Named Entity Recognition - Is There a Glass Ceiling?*
Tomasz Stanislawek, Anna Wróblewska, Alicja Wójcicka, Daniel Ziembicki and Przemyslaw Biecek

16:30–18:00 *Low-Rank Approximations of Second-Order Document Representations*
Jarkko Lagus, Janne Sinkkonen and Arto Klami

16:30–18:00 *Named Entity Recognition with Partially Annotated Training Data*
Stephen Mayhew, Snigdha Chaturvedi, Chen-Tse Tsai and Dan Roth

16:30–18:00 *Contextualized Cross-Lingual Event Trigger Extraction with Minimal Resources*
Meryem M'hamdi, Marjorie Freedman and Jonathan May

16:30–18:00 *Deep Structured Neural Network for Event Temporal Relation Extraction*
Rujun Han, I-Hung Hsu, Mu Yang, Aram Galstyan, Ralph Weischedel and Nanyun Peng

16:30–18:00 *Investigating Entity Knowledge in BERT with Simple Neural End-To-End Entity Linking*
Samuel Broscheit

16:30–18:00 *Unsupervised Adversarial Domain Adaptation for Implicit Discourse Relation Classification*
Hsin-Ping Huang and Junyi Jessy Li

16:30–18:00 *Evidence Sentence Extraction for Machine Reading Comprehension*
Hai Wang, Dian Yu, Kai Sun, Jianshu Chen, Dong Yu, David McAllester and Dan Roth

16:30–18:00 *SimVecs: Similarity-Based Vectors for Utterance Representation in Conversational AI Systems*
Ashraf Mahgoub, Youssef Shahin, Riham Mansour and Saurabh Bagchi

16:30–18:00 *Incorporating Interlocutor-Aware Context into Response Generation on Multi-Party Chatbots*
Cao Liu, Kang Liu, Shizhu He, Zaiqing Nie and Jun Zhao

16:30–18:00 *Memory Graph Networks for Explainable Memory-grounded Question Answering*
Seungwhan Moon, Pararth Shah, Anuj Kumar and Rajen Subba

Monday, November 4, 2019 (continued)

16:30–18:00 *TripleNet: Triple Attention Network for Multi-Turn Response Selection in Retrieval-Based Chatbots*
Wentao Ma, Yiming Cui, Nan Shao, Su He, Wei-Nan Zhang, Ting Liu, Shijin Wang and Guoping Hu

16:30–18:00 *Relation Module for Non-Answerable Predictions on Reading Comprehension*
Kevin Huang, Yun Tang, Jing Huang, Xiaodong He and Bowen Zhou

16:30–18:00 *Slot Tagging for Task Oriented Spoken Language Understanding in Human-to-Human Conversation Scenarios*
Kunho Kim, Rahul Jha, Kyle Williams, Alex Marin and Imed Zitouni

16:30–18:00 *Window-Based Neural Tagging for Shallow Discourse Argument Labeling*
René Knaebel, Manfred Stede and Sebastian Stober

16:30–18:00 *TILM: Neural Language Models with Evolving Topical Influence*
Shubhra Kanti Karmaker Santu, Kalyan Veeramachaneni and Chengxiang Zhai

16:30–18:00 *Pretraining-Based Natural Language Generation for Text Summarization*
Haoyu Zhang, Jingjing Cai, Jianjun Xu and Ji Wang

16:30–18:00 *Goal-Embedded Dual Hierarchical Model for Task-Oriented Dialogue Generation*
Yi-An Lai, Arshit Gupta and Yi Zhang

16:30–18:00 *Putting the Horse before the Cart: A Generator-Evaluator Framework for Question Generation from Text*
Vishwajeet Kumar, Ganesh Ramakrishnan and Yuan-Fang Li

16:30–18:00 *In Conclusion Not Repetition: Comprehensive Abstractive Summarization with Diversified Attention Based on Determinantal Point Processes*
Lei Li, Wei Liu, Marina Litvak, Natalia Vanetik and Zuying Huang

16:30–18:00 *Generating Formality-Tuned Summaries Using Input-Dependent Rewards*
Kushal Chawla, Balaji Vasan Srinivasan and Niyati Chhaya

16:30–18:00 *Do Massively Pretrained Language Models Make Better Storytellers?*
Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola and Christopher D. Manning

16:30–18:00 *Self-Adaptive Scaling for Learnable Residual Structure*
Fenglin Liu, Meng Gao, Yuanxin Liu and Kai Lei

Monday, November 4, 2019 (continued)

16:30–18:00 *BIOfid Dataset: Publishing a German Gold Standard for Named Entity Recognition in Historical Biodiversity Literature*
Sajawel Ahmed, Manuel Stoeckel, Christine Driller, Adrian Pachzelt and Alexander Mehler

16:30–18:00 *Slang Detection and Identification*
Zhengqi Pei, Zhewei Sun and Yang Xu

16:30–18:00 *Alleviating Sequence Information Loss with Data Overlapping and Prime Batch Sizes*
Noémien Kocher, Christian Scuito, Lorenzo Tarantino, Alexandros Lazaridis, Andreas Fischer and Claudiu Musat

16:30–18:00 *Global Autoregressive Models for Data-Efficient Sequence Learning*
Tetiana Parshakova, Jean-Marc Andreoli and Marc Dymetman

16:30–18:00 *Learning Analogy-Preserving Sentence Embeddings for Answer Selection*
Aïssatou Diallo, Markus Zopf and Johannes Fürnkranz

16:30–18:00 *A Simple and Effective Method for Injecting Word-Level Information into Character-Aware Neural Language Models*
Yukun Feng, Hidetaka Kamigaito, Hiroya Takamura and Manabu Okumura

16:30–18:00 *On Model Stability as a Function of Random Seed*
Pranava Madhyastha and Rishabh Jain

16:30–18:00 *Studying Generalisability across Abusive Language Detection Datasets*
Steve Durairaj Swamy, Anupam Jamatia and Björn Gambäck

16:30–18:00 *Reduce & Attribute: Two-Step Authorship Attribution for Large-Scale Problems*
Michael Tschuggnall, Benjamin Murauer and Günther Specht

16:30–18:00 *Variational Semi-Supervised Aspect-Term Sentiment Analysis via Transformer*
Xingyi Cheng, Weidi Xu, Taifeng Wang, Wei Chu, Weipeng Huang, Kunlong Chen and Junfeng Hu

16:30–18:00 *Learning to Detect Opinion Snippet for Aspect-Based Sentiment Analysis*
Mengting Hu, Shiwan Zhao, Honglei Guo, Renhong Cheng and Zhong Su

16:30–18:00 *Multi-Level Sentiment Analysis of PolEmo 2.0: Extended Corpus of Multi-Domain Consumer Reviews*
Jan Kocoń, Piotr Miłkowski and Monika Zaśko-Zielińska

Monday, November 4, 2019 (continued)

16:30–18:00 *A Personalized Sentiment Model with Textual and Contextual Information*
Siwen Guo, Sviatlana Höhn and Christoph Schommer

16:30–18:00 *Cluster-Gated Convolutional Neural Network for Short Text Classification*
Haidong Zhang, Wancheng Ni, Meijing Zhao and Ziqi Lin

16:30–18:00 *Coherence-Based Modeling of Clinical Concepts Inferred from Heterogeneous Clinical Notes for ICU Patient Risk Stratification*
Tushaar Gangavarapu, Gokul S Krishnan and Sowmya Kamath

16:30–18:00 *Predicting the Role of Political Trolls in Social Media*
Atanas Atanasov, Gianmarco De Francisci Morales and Preslav Nakov

16:30–18:00 *Towards a Unified End-to-End Approach for Fully Unsupervised Cross-Lingual Sentiment Analysis*
Yanlin Feng and Xiaojun Wan

Invited Talk I

Ecological Language: A Multimodal Approach to the Study of Human Language Learning and Processing

Gabriella Vigliocco
Department of Experimental Psychology, University College London, UK

Abstract

The human brain has evolved the ability to support communication in complex and dynamic environments. In such environments, language is learned, and mostly used in face-to-face contexts in which processing and learning are based on multiple cues both linguistic and non-linguistic (such as gestures, eye gaze, mouth patterns and prosody). Yet, our understanding of how language is learnt and processed - as well as applications of this knowledge - comes mostly from reductionist approaches in which the multimodal signal is reduced to speech or text. I will introduce our current programme of research that investigates language in real-world settings in which the listener/learner has access to – and therefore can take advantage of – the multiple cues provided by the speaker. I will then describe studies that aim at characterising the distribution of the multimodal cues in the language used by caregivers when interacting with their children (mostly 2-4 years old) and provide data concerning how these cues are differentially distributed depending upon whether the child knows the objects being talked about (allowing us to more clearly isolate learning episodes), and whether objects being talked about are present. I will then move to a study using EEG addressing the question of how discourse but crucially also the non-linguistic cues modulate predictions about the next word in a sentence. Throughout the talk, I will highlight the ways in which this real world, more ecologically valid, approach to the study of language bear promise across disciplines.

Biography

Gabriella Vigliocco is Professor of the Psychology of Language in the Department of Experimental Psychology at University College London, Royal Society Wolfson Research Merit Fellow and Director of the Leverhulme Doctoral training Programme for the Ecological Study of the Brain. She received her PhD from University of Trieste in 1995, was a post-doc at University of Arizona, and after being at University of Wisconsin as Assistant Professor and the Max Planck Institute for Psycholinguistics as a visiting scientist, she moved to UCL. Vigliocco leads a multidisciplinary team composed of psychologists, linguists, computer scientists and cognitive neuroscientists sharing the vision that understanding language and cognition requires integration of multiple levels of analysis and methodological approaches. Her research focuses on the cognitive and neurobiological basis of human communication. More specifically she is interested in how we learn and process language in real-word settings, how our semantic knowledge interfaces with perception, action and emotion and how these systems are recruited during language learning. Through the years, her work has been supported by numerous prestigious awards, including Human Frontier Science Programme and currently European Research Council.

Invited Talk II

Multi-Step Reasoning for Answering Complex Questions

Christopher Manning
Department of Linguists and Computer Science, Stanford University, USA

Abstract

Current neural network systems have had enormous success on matching but still struggle in supporting multi-step inference. In this talk, I will examine two recent lines of work to address this gap, done with Drew Hudson and Peng Qi. In one line of work we have developed neural networks with explicit structure to support attention, composition, and reasoning, with an explicitly iterative inference architecture. Our Neural State Machine design also emphasizes the use of a more symbolic form of internal computation, represented as attention over symbols, which have distributed representations. Such designs encourage modularity and generalization from limited data. We show the model's effectiveness on visual question answering datasets. The second line of work makes progress in doing multi-step question answering over a large open-domain text collection. Most previous work on open-domain question answering employs a retrieve-and-read strategy, which fails when the question requires complex reasoning, because simply retrieving with the question seldom yields all necessary supporting facts. I present a model for explainable multi-hop reasoning in open-domain QA that iterates between finding supporting facts and reading the retrieved context. This GoldEn Retriever model is not only explainable but shows strong performance on the recent HotpotQA dataset for multi-step reasoning.

Biography

Christopher Manning is the inaugural Thomas M. Siebel Professor in Machine Learning in the Departments of Computer Science and Linguistics at Stanford University and Director of the Stanford Artificial Intelligence Laboratory (SAIL). His research goal is computers that can intelligently process, understand, and generate human language material. Manning is a leader in applying Deep Learning to Natural Language Processing, with well-known research on Tree Recursive Neural Networks, the GloVe model of word vectors, sentiment analysis, neural network dependency parsing, neural machine translation, question answering, and deep language understanding. He also focuses on computational linguistic approaches to parsing, robust textual inference and multilingual language processing, including being a principal developer of Stanford Dependencies and Universal Dependencies. He is an ACM Fellow, a AAAI Fellow, and an ACL Fellow, and a Past President of the ACL (2015). His research has won ACL, Coling, EMNLP, and CHI Best Paper Awards. He has a B.A. (Hons) from The Australian National University and a Ph.D. from Stanford in 1994, and he held faculty positions at Carnegie Mellon University and the University of Sydney before returning to Stanford. He is the founder of the Stanford NLP group (@stanfordnlp) and manages development of the Stanford CoreNLP software.

Analysing Neural Language Models: Contextual Decomposition Reveals Default Reasoning in Number and Gender Assignment

Jaap Jumelet
jumeletjaap@gmail.com
University of Amsterdam

Willem Zuidema
w.h.zuidema@uva.nl
ILLC, University of Amsterdam

Dieuwke Hupkes
d.hupkes@uva.nl
ILLC, University of Amsterdam

Abstract

Extensive research has recently shown that recurrent neural language models are able to process a wide range of grammatical phenomena. *How* these models are able to perform these remarkable feats so well, however, is still an open question. To gain more insight into what information LSTMs base their decisions on, we propose a *generalisation* of *Contextual Decomposition* (GCD). In particular, this setup enables us to accurately distil which part of a prediction stems from semantic heuristics, which part truly emanates from syntactic cues and which part arise from the model biases themselves instead. We investigate this technique on tasks pertaining to syntactic agreement and co-reference resolution and discover that the model strongly relies on a *default reasoning* effect to perform these tasks.

1 Introduction

Modern language models that use deep learning architectures such as LSTMs, bi-LSTMs and Transformers, have shown enormous gains in performance in the last few years and are finding applications in novel domains, ranging from speech recognition and writing assistance to autonomous generation of fake news. Understanding how they reach their predictions has become a key question for NLP, not only for purely scientific, but also for practical and ethical reasons.

From a linguistic perspective, a natural approach is to test the extent to which these models have learned classical linguistic constructs, such as inflectional morphology, constituency structure, agreement between verb and subject, filler-gap dependencies, negative polarity or reflexive anaphora. An influential paper using this approach was presented by Linzen et al. (2016), who investigated the performance of an LSTM-based language model on number agreement. In many later papers (e.g. Gulordava et al., 2018; Wilcox et al., 2018; Jumelet and Hupkes, 2018; Marvin and Linzen, 2018; Giulianelli et al., 2018) a wide spectrum of grammatical phenomena has been investigated, assessing these grammatical abilities in a mainly "behavioural" fashion, by considering the model's output.

In this paper, we take it as established that neural language models have indeed learned a great number of non-trivial linguistic patterns and ask instead *how* language models come to show this behaviour, and, more specifically, what kind of information they use to come to their decisions. There exist already a number of approaches that look inside the high-dimensional vector representations and non-linear functions of these models, trying to track the flow of information. In the next section, we will review some of that work, distinguishing between hypothesis-driven and data-driven methods. We highlight in particular one method called Contextual Decomposition (CD, Murdoch et al., 2018), that combines the strengths of hypothesis- and data-driven analysis methods.

In the remainder of this paper, we then propose a generalisation of this method, which we call Generalised Contextual Decomposition ("GCD"). We derive equations for GCD for the case of a unidirectional (one or multi-layer) LSTM (Hochreiter and Schmidhuber, 1997), and use the method to analyse how a language model processes two different phenomena: number agreement and gendered pronoun resolution.

We demonstrate the power of GCD through the revelation of some important asymmetries in the way that both the singular-plural and the male-female distinction are handled. In particular, we find evidence for a *default reasoning* effect, which we believe could also be important for future work on detecting and removing bias: a default category (singular, masculine) appears to be hard-coded in

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 1–11
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

the weights of the language model, number and gender information in the word embeddings themselves mainly plays a role for phrases of the opposite category (plural, feminine). Furthermore, GCD enables us to investigate pronoun resolution in a way that has not been done before: by delving into the model reasoning we are able to accurately pinpoint where and how this resolution takes place.[1]

2 Network analysis methods

Recently, methods to open the blackbox of deep neural networks have become an important research area (see Poerner et al., 2018; Belinkov and Glass, 2019, for recent reviews of proposed methods in NLP). We distinguish between hypothesis-driven methods, and data-driven methods. Hypothesis-driven methods include probes or *diagnostic classifiers*, that test whether specific, a priori defined information can be decoded from the internal states of a neural model, many ablation studies, and types of correlation analysis, where correlations between the structure of internal representations of better and lesser understood models are studied). An example of this approach is Giulianelli et al. (2018), who trained linear diagnostic classifiers on all layers and gate activations of an LSTM to predict the number of the subject that the verb, occurring later in the sentence, needs to agree with (i.e. the number-agreement task). Their results show that the relevant information is encoded in a different way in different components of the model, and at different times while processing a sentence. This result is interesting, because it starts from a clearly interpretable hypothesis (number information must be maintained somewhere while the network traverses the sentence), but the work also demonstrates the limitations of the approach: It progresses one hypothesis about one linguistic pattern at a time and involves much training, work, and computation at each step.

Data-driven methods include gradient-based methods and contextual decomposition. An example of a gradient-based method is Arras et al. (2017), who adapt Layer-wise Relevance Propagation (LRP, Bach et al., 2015) to the case of LSTMs. The key idea is to run the LSTM on each input of interest (the forward pass), then define a relevance vector at the output layer and propagate that relevance backwards through the network. The relevance vector simply singles out the dimensions of the output of interest, and sets all other dimension to zero. The backward pass is almost standard backpropagation, except that relevance does not backpropagate into the gates.While Arras et al.'s results reveal interesting patterns in sentences used in a sentiment classification task, their work illustrates some limitations as well. In particular, the work deals with a classification task with few classes, aggregates relevance per word for each predicted class, but offers little insight in how word meanings interact to build up sentence meaning beyond 'pushing in the right direction' vs. 'pushing in the wrong direction'.

An alternative data driven method, and the one that we will expand on in this paper, is Contextual Decomposition for LSTMs (CD, Murdoch et al., 2018). The key idea behind this technique is to partition the hidden states into two components, that Murdoch et al. label 'relevant' and 'irrelevant'. For each word in a sentence, they do a forward pass that computes all cell and gate activations as in normal operation of the neural network, but also partition each activation value of each neuron in h or c in a part that is *caused* by some selected token or phrase in focus, and a part that is not. They achieve this by deriving a factorisation of the update formulas for h and c, that expresses them as a long sum of components and then selecting some of these components as being relevant, and others as irrelevant. Qualitative results on sentiment analysis suggest that CD can attribute roles to words in a sentence very well, better than alternatives the authors considered (which, unfortunately, did not include LRP).

CD thus requires no extra training and requires only the forward pass of the network. It can easily be extended to work efficiently with many classes, such as the language modelling task that we are interested in. In the next section, we will define CD more precisely, where we will use the terms *inside* and *outside* rather than relevant and irrelevant. We then propose a generalisation that allows us to experiment with different *hypotheses* on what goes into the *inside* and *outside* bins, enabling some of the advantages of hypothesis-driven analysis methods to be brought into this data-driven method.

[1] We have integrated all our code in `diagnnose` (Jumelet and Hupkes, 2019), a well-documented analysis library which facilitate the diagnosis of neural network activations: `github.com/i-machine-think/diagnnose`.

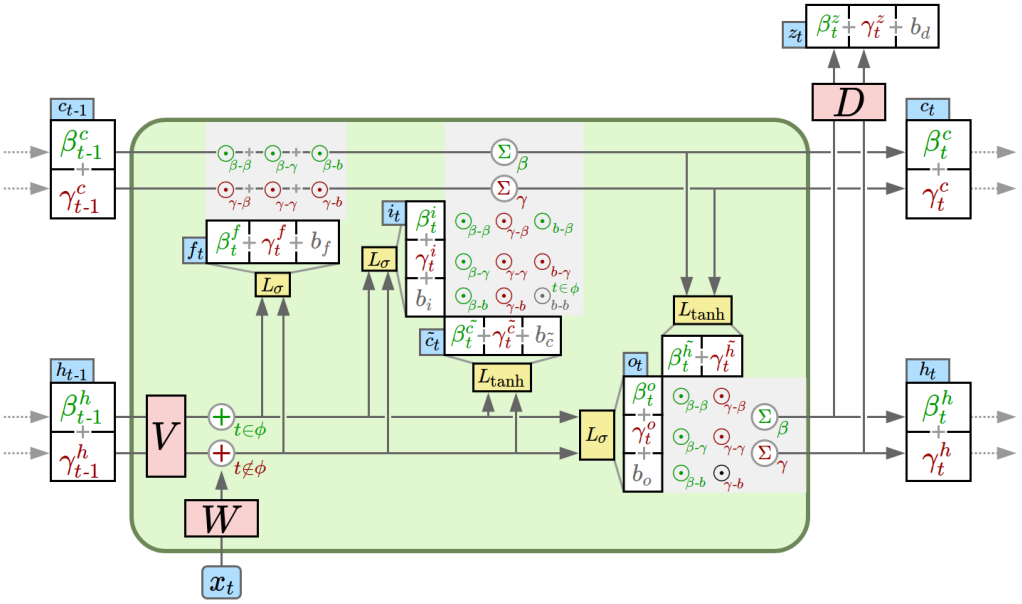


Figure 1: A graphical overview of GCD, based on the LSTM design of Olah (2015). ϕ denotes the phrase in focus, and $t \in \phi$ implies the action is only performed when step t is part of ϕ. $\odot$ denotes an individual interaction; green interactions are added the β part and red interactions to γ. V, W, and D represent the linear projections of the LSTM itself. The interaction set denoted here corresponds to the IN set of Equation 12.

3 Generalised Contextual Decomposition

In this particular study, we consider the LSTM language model that was made available by Gulordava et al. (2018). This language model (LM) is a 2-layer LSTM with 650 hidden units in both layers, trained on a corpus with Wikipedia data. Given the relevance of the specific LSTM-dynamics for the understanding of the main method of our paper, we repeat the equations that describe it below.

$$f_t = \sigma(W_f x_t + V_f h_{t-1} + b_f) \quad (1)$$
$$i_t = \sigma(W_i x_t + V_i h_{t-1} + b_i) \quad (2)$$
$$\tilde{c}_t = \tanh(W_{\tilde{c}} x_t + V_{\tilde{c}} h_{t-1} + b_{\tilde{c}}) \quad (3)$$
$$o_t = \sigma(W_o x_t + V_o h_{t-1} + b_o) \quad (4)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (5)$$
$$h_t = o_t \odot \tanh(c_t) \quad (6)$$
$$z_t = D h_t + b_d \quad (7)$$
$$p_t = \text{SoftMax}\,(z_t) \quad (8)$$

The final model output p_t represents a multinomial distribution over the model's vocabulary. Throughout the paper we refer to the bias terms b as the model *intercepts*, to avoid confusion with general biases that the model may have.

CD To compute the contributions of one or multiple input tokens (said to be *in focus*) to the output of an LSTM cell, Murdoch et al. (2018) divide each cell and hidden state into a sum of two parts: a β part, which contains the part of this particular state that stems from **inside** this phrase, and a γ part, which contains information coming from words **outside** this phrase. The output logit z_t can then be redefined as

$$\begin{aligned} z_t = D h_t + b_d &= D\beta_t^h + D\gamma_t^h + b_d \\ &= \beta_t^z + \gamma_t^z + b_d \end{aligned}$$

with β_t^z providing a quantitative score of the phrase's contribution to the logit. How a particular hidden state h_t is partitioned into β_t^h and γ_t^h is determined by two things: **i)** The decomposition of the *previous* states c_{t-1} (β_{t-1}^c and γ_{t-1}^c) and h_{t-1} (β_{t-1}^h and γ_{t-1}^h), and **ii)** Which *interactions* between the different β and γ terms, the intercepts b, and the input x_t are considered to be part of the inside contribution of the phrase. We provide a graphical overview of our setup in Figure 1.

Factorised activation functions The gate interactions cannot yet be expanded into a cross-term of their input parts, due to the non-linear activation that wraps them. Murdoch et al. define a method to factorise the sigmoid and tanh functions for each specific gate into a sum of contributions of the input terms, such that

$$\tanh(\sum_{i=1}^{N} y_i) = \sum_{i=1}^{N} L_{\tanh}(y_i)$$

$L_{\tanh}$ expresses the *contribution* of each input, which is computed by averaging over the differences of all possible permutations of the input

terms; a procedure that corresponds to the calculation of the Shapley values (Shapley, 1953).[2]

Before this factorisation is performed, an input token x_t is added to the inside part β if it is part of the phrase for which we decompose (i.e. the phrase *in focus*), otherwise it is added to γ. Equation 1, for example, can then be rewritten as:

$$\begin{aligned} f_t &= \sigma\left(V_f\beta^h_{t-1} + V_f\gamma^h_{t-1} + W_f x_t + b_f\right) \\ &= L_\sigma\left(V_f\beta^h_{t-1} + W_f x_t\right) + L_\sigma(\gamma^h_{t-1}) + L_\sigma(b_f) \end{aligned} \tag{9}$$

where x_t is considered to be inside the phrase in focus and therefore added to the β part (denoted in green for extra emphasis). A similar sum can be written down for the input gate i_t and the candidate cell state $\tilde{c}_t$. This allows the two products $f_t \odot c_t$ and $i_t \odot \tilde{c}_t$ of Equation 5 to be expanded into a sum of cross-terms between the decomposed gate and (candidate) cell values. Expanding the forget and input gate results in 15 cross-terms, that each express different interactions between the current input, previous β and γ terms, and the model intercepts.

Murdoch et al. state they observed improvements when the intercept term is fixed to the first position in each permutation. Consequently, however, these intercepts are assigned a relatively larger contribution, as their fixed position makes their contribution independent of the magnitudes of the other terms. We therefore pose that the full set of permutations should be considered, to assign unbiased contributions to each input term.[3]

Decomposing interactions Based on all the different interaction terms, the decomposition is determined by *which* of these interactions should be considered to belong to the inside part β of the next cell state and which to the outside part γ.

In the formulation of Murdoch et al., all interactions with outside parts γ_t are disregarded for the computation of β_{t+1}, and therefore only information directly stemming from the β_t terms with no interference from γ_t is taken into account. Of the 15 cross-product terms described above, this leaves 5 terms to be part of β^c_{t+1}:

$$\begin{aligned} \beta^c_{t+1} = {} & L_\sigma(V_f\beta^h_t + W_f x_t) \odot \beta^c_t && \beta\text{-}\beta \\ & + L_\sigma(b_f) \odot \beta^c_t && \beta\text{-}b \\ & + L_\sigma(V_i\beta^h_t + W_i x_t) \odot L_{\tanh}(V_{\tilde{c}}\beta^h_t + W_{\tilde{c}} x_t) && \beta\text{-}\beta \\ & + L_\sigma(V_i\beta^h_t + W_i x_t) \odot L_{\tanh}(b_{\tilde{c}}) && \beta\text{-}b \\ & + L_{\tanh}(V_{\tilde{c}}\beta^h_t + W_{\tilde{c}} x_t) \odot L_\sigma(b_i) && \beta\text{-}b \end{aligned} \tag{10}$$

The remaining 10 terms from the cross-product are put in γ^c_{t+1}. We use the notation $\{\beta\text{-}\beta,\ \beta\text{-}b\}$ to concisely describe this set of interactions. The decomposition of the hidden state is created by decomposing the output gate:

$$\begin{aligned} \beta^h_{t+1} = {} & L_\sigma(V_o\beta^h_t + W_o x_t) \odot \beta^c_{t+1} \\ & + L_\sigma(b_o) \odot \beta^c_{t+1} \end{aligned} \tag{11}$$

The decomposed contribution score β^z_T over the model vocabulary at step T of some phrase in focus is then calculated by passing the decomposed hidden state to the decoder, i.e. $D\beta^h_T$. This score can be expressed as a relative contribution by normalising it by the full model logit z (including b_d). In a multi-layer LSTM, β and γ parts are not only propagated *forward*, but also *upward*, where they are added to their respective parts in the layer above them. For initialisation β is set to a zero vector, and γ is set to the initial LSTM states.[4]

Generalising CD While Murdoch et al. (2018) consider only one way of partitioning interactions between inside and outside components, their setup can be quite easily generalised to also allow other interactions to be included in the inside terms β. To obtain a better insight into how different interactions contribute to the final prediction, we experiment with various ways of defining the set of relevant interactions.

A particular case concerns the interactions between β and γ. It wouldn't be correct to completely attribute the information flowing from these interactions to the phrase in focus, but disallowing any information stemming from interactions of a phrase with a subsequent token results in loss of relevant information. Consider, for instance, the verb prediction in a number agreement task. While the correct verb *form* depends only on the subject, the right *time* for this information to surface depends on the material in between, which in the setup described in Equation 10 would be discarded by assigning the β-γ interactions to γ.

[2] In the original formulation this procedure is called *linearizing*. We deemed this term to be slightly confusing, as the resulting functions L are still non-linear.

[3] We only discovered the impact of this decision after the paper had already been reviewed. While using the full set of permutations did, fortunately, not qualitatively change our conclusions, the exact numbers presented in this work thus differ from the earlier version of this paper. For completeness, we report the original results with the fixed intercept positions in the supplementary materials of this article.

[4] For the initial states we use the activations that follow from the short phrase ". <eos>". This phrase resets the model state to a clean slate, and leads to better results than using 0-valued activations.

Taking inspiration from Arras et al. (2017), and based on their motivation, we add the β_s-γ_g interaction to the relevant interaction set, while still disregarding a γ_s-β_g interaction. The g subscript denotes the part of the interaction that is coming from the gate, and s the source part. We denote this amended interaction as β-γ^*.

Furthermore, we follow the addition of Singh et al. (2019) of only adding the intercept interactions b-b to the inside part if the current time step is part of the phrase in focus, which we denote as $b\text{-}b \in x$. We add these β-γ^* and $b\text{-}b \in x$ interactions to Equation 10, resulting in the following decomposition that is presumed to come from **inside** the phrase in focus (denoted as IN):

$$\begin{aligned}\beta^c_{t+1} = \{&\beta\text{-}\beta, \beta\text{-}b\} \\ &+ L_\sigma(V_f\gamma^h_t) \odot \beta^c_t && \beta\text{-}\gamma^* \\ &+ L_\sigma(V_i\gamma^h_t) \odot L_{\tanh}(V_{\tilde{c}}\beta^h_t + W_{\tilde{c}}x_t) && \beta\text{-}\gamma^* \\ &+ L_\sigma(b_i) \odot L_{\tanh}(b_{\tilde{c}}) && b\text{-}b \in x \quad (12)\end{aligned}$$

We also experimented with various other interaction sets. To determine the influence of the gate intercepts, we create an interaction set that does not take the input embeddings into account at all: $\{\beta\text{-}\beta, \beta\text{-}\gamma^*, \beta\text{-}b, b\text{-}b\}$, with x always added to γ, denoted as INTERCEPT*. We include β-γ^* to still account for the way the intercepts are gated by the input sentence. The initial hidden and cell state are added to β now as well, as we consider these states to be part of the model bias. Finally, to determine the dependence of the input on the gate intercepts we use an interaction set that never takes the interactions with any intercept into account: $\{\beta\text{-}\beta, \beta\text{-}\gamma^*\}$, denoted as $\neg$INTERCEPT.

4 Experimental setup

We use GCD to study how our LSTM model handles two different linguistic phenomena: subject-verb agreement and anaphora resolution in relation to *gender*. Next to the model of Gulordava et al. (for which we present our results), we also ran our experiments on the LM of Józefowicz et al. (2016), which arrives at similar results.

4.1 Subject-verb agreement

We consider a variant of the *number-agreement* (NA) task that was proposed by Linzen et al. (2016) to assess the syntax-sensitivity of language models. In this task, a model is evaluated based on its ability to track a long-distance subject-verb relation, which is assessed by the percentage of times that the verb-form it prefers matches the *number* of the syntactic subject. Commonly, the material in between subject and verb contains an *attractor* noun that competes with the syntactic subject, e.g. *The **keys** on the table **are***.

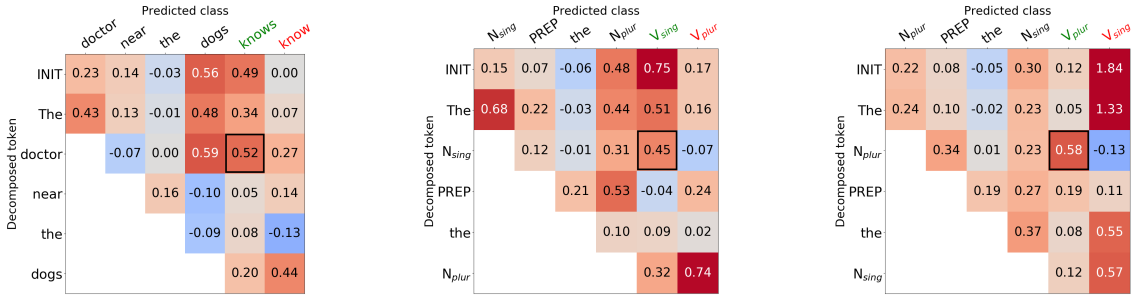
Here, we consider the NA corpora made available by Lakretz et al. (2019), which consists of a number of data sets containing a range of syntactic constructions in which number agreement plays a role. We report results for several of their data sets, but focus in particular on their *NounPP* subset, in which sentences contain an attractor embedded in a prepositional phrase. These sentences are formed following the template *The* `N Prep` *the* `N V` [..], e.g. *The boys near the car greet* [..]. The sentences in this data set are split based on the number of the subject and the attractor, resulting in four different conditions: SS, SP, PS, and PP.

4.2 Anaphora resolution and gender bias

Our second experiment concerns anaphora resolution and the possible gender biases that networks may use to perform this task. We focus on intra-sentential anaphora resolution, in which a pronoun in a subordinate clause refers to an entity in the main clause, based on *gender information*. For example: *The monk liked the **nun**, because **she** was always nice to him*.

Compared to number agreement it is more difficult to formulate a setup for anaphora resolution in which there is a right or wrong prediction that directly reflects how the model handles the phenomenon: when predicting ***she*** in the example, it could have been equally probable to predict ***he***. Rather, to establish if a model correctly resolves the referent of a pronoun, it should be checked what the model considered to be the source of this prediction, which cannot directly be inferred from the prediction itself. GCD gives us exactly this information and is therefore an excellent tool to study anaphora resolution in language modeling.

To create our corpus, we use the templates from the WinoBias corpus created by Zhao et al. (2018). This corpus contains sentences with job titles that are gender neutral, yet contain a stereotypical bias towards one gender (doctors and CEOs are *male*, nurses and housekeepers *female*). We construct two types of corpora, one containing the stereotypical job titles of Zhao et al. and one in which we replace these titles by entity descriptions that are unambiguously gendered (*king*, *bride*, *father*, etc.). Similar to the *NounPP* corpus for NA, we



(a) A single *NounPP* sentence: singular subject with plural attractor (SP).

(b) Average *NounPP* SP: singular subject with plural attractor.

(c) Average *NounPP* PS: plural subject with singular attractor.

Figure 2: Average contributions for the NounPP corpus of Lakretz et al. (2019), defined as β_t^z / z_t. INIT denotes the contribution of the initial states. The picture depicts an asymmetry in the way that the model encodes singularity and plurality: while plural verbs depend strongly on the subject, for singular sentences this is not the case.

create 4 different conditions, based on the gender of the subject and object (FF, FM, MF, and MM). An example of an MF sentence would be *The father likes the woman, because he/she*. We sample from the set of entity descriptions to create 500 sentences per condition, for both corpus types.

4.3 Experiment types

Phrase contributions In the first type of experiment, we consider the contributions of different words in the input to a later prediction of the model. This allows us to compare the contributions of different words in the sentence and track which words the model uses to come to its prediction. We compute a phrase's contribution to a prediction at step t as β_t^z / z_t.

Pruning information In the second type of experiment, we focus on the model's *predictions*. In particular, we study how the model's predictions change when it is forced to consider only specific parts of the input, by disregarding all information that does not belong to the inside information of that part of the input. This allows us to quantify the extent to which a correct prediction does in fact stem from that phrase. For this experiment, we consider several different interaction sets, that differ in what is considered to be inside the contribution of the phrase: IN describes the direct contribution of some phrase, INTERCEPT the contribution of the model intercepts, and ¬INTERCEPT the contribution of some phrase without its intercept interactions.

5 Subject-verb Agreement

We now study what information the LM uses to achieve the high prediction accuracies that were reported by Lakretz et al. (2019).

5.1 Phrase contributions

For every word in a sentence, we compute the GCD contribution for all words preceding this word. We plot these contributions in a *decomposition matrix* (akin to the attention plots often seen in machine translation papers). Every cell of this matrix represents the contribution of an input x_i (row i) to an output y_j (column j). The complete decomposition of an output word y_j can thus be found in column j. The reported scores are the decomposed scores normalised by the total model logit, resulting in the relative contribution.

In Figure 2, we plot the average decomposition matrices for the SP and PS splits of the NounPP data set. While many interesting observations can be made here, we would like to focus on the final 2 columns that represent the decompositions of the correct and wrong verb in the sentence, and on the contribution of the subject to this verb. In the singular case (2b), this contribution is, surprisingly, relatively low: The correct verb prediction does not seem to depend solely on the syntactic subject, but stems from elements that lie outside the subject as well. For the plural case, this picture is strikingly different: The highest contribution now stems from the subject of the sentence. When considering the decomposition of the wrong verb (the final column) it becomes even more clear that contributions to a plural verb predominantly stem from a plural noun, whereas singular verbs receive strong contributions from non-numbered tokens as well. This quite remarkable difference provides the first evidence for one of our conclusions: A singular prediction acts as the default number for the model, and predicting a plural verb requires

			GCD		
Task	**C**	FULL	IN	INTERCEPT*	¬INTERCEPT
Simple	S	100	73.3 (91.3)	97.3 (100)	69.7 (86.3)
Simple	P	100	100 (100)	32.7 (7.7)	100 (100)
nounPP	SS	99.2	93.0 (99.7)	99.8 (99.8)	72.7 (88.7)
nounPP	SP	87.2	90.3 (99.3)	98.8 (99.8)	60.5 (83.5)
nounPP	PS	92.0	100 (100)	0.0 (0.0)	100 (100)
nounPP	PP	99.0	100 (99.3)	7.0 (0.5)	99.8 (100)
namePP	SS	99.3	97.7 (91.3)	99.4 (100)	76.2 (90.9)
namePP	PS	68.9	98.3 (98.2)	1.3 (0.0)	99.9 (99.9)

Table 1: Accuracies on various subject-verb agreement tasks of Lakretz et al. (2019). FULL denotes the full model accuracies. IN is the decomposition of the subject, INTERCEPT* only decomposes the gate intercepts of the model. ¬INTERCEPT takes no interactions with the intercepts into account. Singular conditions are denoted in green. $(\cdot)$ denotes accuracies of scores without decoder bias, i.e. Dh_t vs $Dh_t + b_d$.

some explicit evidence coming from the subject.

5.2 Pruning information

To quantify to which extent the model bases its prediction on the subject, we prune all information that is not directly related to the subject and repeat Lakretz et al.'s NA tasks. If the model prediction were based solely on the number of the subject, its accuracy should go up, as we filter out all potentially intervening or confusing information. If, on the other hand, the prediction of the verb is not causally linked to the subject, but the model is using heuristics that require the rest of the sentence, no increase in accuracy is to be expected. We show the results, along with the accuracy of the full model in Table 1.

These numbers show a strong causal relation between plural subjects and verbs: The number prediction accuracy for the IN decomposition goes up for all cases with a plural subject. This confirms our previous finding from the decomposition matrix, which showed a relatively high contribution of plural subjects to plural verbs, as well as the conclusion of Lakretz et al. (2019) that the model is in fact keeping track of syntactic structure.

When considering the singular subjects an interesting pattern emerges: The decomposition of sentences for which the intervening attractor has the same number leads to a *lower* accuracy. This confirms that the model is in fact basing its prediction for these conditions on information that lies outside the subject itself.

Intercepts When we only decompose with respect to the gate intercepts (INTERCEPT*, column 5) it turns out the model has an extreme preference for selecting singular verbs. Decomposing without the intercept interactions (NO INTERCEPT, column 6) leads (as expected) to opposite results: the decomposed model now has a strong preference towards plural verbs as the singular prediction no longer can depend on these intercepts. This further confirms that singular verbs are used as a default baseline, which is partly encoded in its intercepts. To predict plural verbs, on the other hand, some evidence is needed, which the model picks up correctly from the subject number.

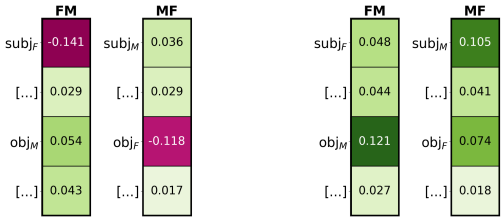
Corpus frequency One would expect that due to the model's default number being singular, this class to be more encountered during training. This turns out not to be the case: in the model's training corpus the plural verbs of the NA tasks occurred over 5 times as often as their singular counterparts. This higher frequency is in fact represented in the decoder intercept, which is higher on average for plural verbs, but it is surprising that the LSTM weights encode a default for the minority class.

6 Anaphora-resolution and gender

For the NA-tasks, the full model accuracy provides evidence that the model can perform the task well; for anaphora resolution, it is not possible to create such accuracies based on the full model predictions alone. In this section, we therefore address two different questions: 1) *Does the model correctly resolve referents?* In other words: When the model generates a male or female pronoun, does it consistently do this based on male and female referents encountered earlier in the sentence, and 2) If the model correctly performs anaphora resolution, what types of interactions and information does it use to do so? In our analysis we furthermore consider the difference between sentences with unambiguously gendered referents with sentences in which the gender of the referents is ambiguous but contains a stereotypical male or female bias.

6.1 Phrase contributions

As the template that the sentences in our anaphora data set follow is not as rigid as those of the NA tasks, creating an averaged decomposition matrix for all words in the sentences does not result in a comprehensive picture. To evaluate whether the model links pronouns to referents of the correct gender, we subtract the referent contribution to *she* from that to *he*: $\beta^z_{he}/z_{he} - \beta^z_{she}/z_{she}$. A positive



(a) unambiguous (b) stereotypical

Figure 3: Average decomposed preference of *he* over *she*, calculated as the difference between the relative contributions: $\beta^z_{he}/z_{he} - \beta^z_{she}/z_{she}$. Positive values denote male preference, negative values female preference. Phrases occurring between subject and object, and object and pronoun are denoted with `[...]`.

difference then indicates this referent had a greater contribution towards predicting *he* than *she*, and a negative difference vice versa. Little difference indicates that the referent did not contribute much to the gender of the predicted pronoun.

Unambiguous referents In Figure 3a we plot this relative contribution difference for the two conditions in our data set that contain both an unambiguous female and male referent. It is evident that the model bases its prediction on a referent of the right gender: The female subjects and objects contribute more to the prediction of *she* (reflected by the negative purple cells) and the male subjects and objects more to the prediction of *he* (the positive green cells).

Interestingly, this effect is much stronger visible for the female connections. The reason for this can be found in the model intercepts; male preference is more strongly encoded in the intercepts of the decoder: *he* has an intercept of 7.75, *she* only 6.09. This enables the model to use this male prediction as a default, similar to how singular verbs acted as a default baseline for number prediction. Akin to number agreement the model thus needs to encounter sufficient evidence of an entity being female to prefer a female pronoun. In the next section we show that this male default is encoded in the gate intercepts as well.

Stereotypical referents The intermediate conclusion that the language model performs successful anaphora resolution on our experiment also provides us the opportunity to probe the gender biases of the model. To do so, we repeat the pronoun preference test on an adapted version of the WinoBias corpus (Zhao et al., 2018), in which all referents are only stereotypically considered to be male or female (e.g., *doctor* and *nurse*). The results, plotted in Figure 3b, show that the model is very susceptible to stereotypically male referents; these decomposed scores contain an even stronger male preference than for the unambiguous corpus. The stereotypically female referents, on the other hand, do not lead to a female preference, indicating that their contribution is not considered strong enough evidence by the model to prefer a female pronoun. All the intermediate tokens exhibit a slight male preference, a pattern that is comparable to the singular bias of the NA task. From these results we conclude that the model considers a stereotypically male job occupation to be male ("*doctors* are male"), whereas this does not hold for stereotypically female jobs.

6.2 Pruning information

Following our subject-verb agreement setup, we compare the predictions of our language model when it focuses only on the subject or object of the sentence. In Table 2, we show the percentage of cases in which *he* is assigned a higher decomposed score than *she*, for both unambiguously gendered referents and stereotypically gendered referents.

FULL In the first column of Table 2a, we see that if the sentence contains referents of the same gender (MM & FF), the full model prediction almost always prefers to use a pronoun with that same gender. When both a male and female referent are present, the model has a slight preference for generating a pronoun that matches with the *subject* of the sentence (which, interestingly, is the referent that is the furthest away from the pronoun). In the stereotypical case (Table 2b), the difference between male and female sentences for the FULL scores almost disappears, showing a predominant male pronoun preference. This shows that the model by default prefers a masculine pronoun, and only when it is provided sufficient evidence of a female entity it will consider predicting *she* (similar to number agreement).

Pruning When considering the decompositions with relation to the subject or object we see that the decomposed score of a male entity in all conditions always prefers a male pronoun. For female entities this effect is slightly obscured by the male bias of the decoder intercept: The accuracies without adding this intercept highlight that female contributions lead to a strong female preference. For the stereotypical corpus this female preference is

		GCD		
	FULL	SUBJECT	OBJECT	INTERCEPT*
MM	100	100 (93.2)	100 (97.8)	100 (93.2)
MF	58.6	100 (86.4)	47.2 (0.8)	100 (96.0)
FM	37.0	29.2 (0.6)	100 (97.2)	100 (98.0)
FF	1.2	77.2 (0.8)	88.8 (1.2)	100 (92.2)

(a) *%he>she*, unambiguous referents

		GCD		
	FULL	SUBJECT	OBJECT	INTERCEPT*
MM	100	100 (100)	100 (100)	100 (88.0)
MF	94.6	100 (99.6)	95.4 (84.0)	100 (84.8)
FM	88.8	90.6 (77.4)	100 (100)	100 (91.0)
FF	84.6	92.8 (75.6)	97.4 (84.0)	100 (89.2)

(b) *%he>she*, stereotypical referents

Table 2: Gender preference on the fixed and stereotypical gender corpora. Reported scores are the percentage of times *he* is preferred over *she*. The first column denotes the gender of the subject and object. FULL denotes the full model preference, SUBJECT the decomposed score of the subject phrase (including determiners), and OBJECT the decomposed object score. INTERCEPT* is the decomposed score with relation to the intercepts only. $(\cdot)$ denotes accuracies of scores without decoder bias, i.e. Dh_t vs $Dh_t + b_d$.

far less apparent, which is in line with the results of Section 6.1. When solely considering the intercept contributions it becomes clear once more that a strong male bias is encoded in them, an effect that is further amplified by the decoder intercept.

Corpus frequency For NA the default class turned out to be less frequent in the training corpus. For our gender setup it turns out the male default is in fact the majority class, with *he* being nearly 4 times more frequent than *she*. We conclude that the default class is not directly correlated to training frequency and likely depends on the phenomenon at hand, although an investigation incorporating a wider range of models would be needed to establish this.

7 Conclusion

We propose a generalised version of Contextual Decomposition (Murdoch et al., 2018) – GCD – that allows to study specifically selected interactions of components in an LSTM language model. This enables GCD to extract the contributions of a model's intercepts, or to investigate the interactions of a phrase with other phrases and intercepts.

We analyse two linguistic phenomena in a pretrained language model: subject-verb agreement, in which *number* plays a role, and anaphora resolution for which *gender* is important. Anaphora resolution in the context of language modelling had not been investigated thoroughly before, and our setup enables this at an unprecedented level.

We trace what information the language model uses to make predictions that require gender and number information and find that, in both cases, the model applies a form of *default reasoning*, by falling back on a default class (male, singular) and predicting a female or plural token only when it is provided enough explicit evidence. As such, the decision to predict masculine and singular words can not be traced back evidently to specific information in the network inputs, but is encoded by default in the model's weights.

Our setup and results demonstrate the power of GCD, which can be applied on top of any model without additional training. Our results bear relevance for work on detecting and removing model biases, and may clarify some of the issues that were raised by Gonen and Goldberg (2019), who argue that current bias removal methods only operate on a superficial level. GCD could also be used to aid a model in guiding it towards the right flow of information, which could be applied to a wide range of applications such as the interventions of Giulianelli et al. (2018). In the future, we plan on extending GCD to other types of language models, such as the currently popular attention-based models. Furthermore, we wish to expand the capacities of GCD by improving the gate factorisation with a better Shapley value approximator, such as those proposed by Lundberg and Lee (2017) or Ancona et al. (2019). The axiomatic approach of Montavon (2019) could provide further insight into how GCD relates to other explanation methods, and we are confident that combining the strengths of GCD with that of other frameworks will ultimately lead to a more *robust* and *faithful* insight into deep neural networks.

Acknowledgements

We thank our anonymous reviewers for their useful suggestions and comments. DH and WZ are funded by the Netherlands Organization for Scientific Research (NWO), through a Gravitation Grant 024.001.006 to the Language in Interaction Consortium.

References

Marco Ancona, Cengiz Öztireli, and Markus Gross. 2019. Explaining deep neural networks with a polynomial time algorithm for shapley values approximation. In *36th International Conference on Machine Learning (ICML 2019)*.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. *EMNLP 2017*, page 159.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the Hood: Using Diagnostic Classifiers to Investigate and Improve how Language Models Track Agreement Information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 609–614.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 1195–1205.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Jaap Jumelet and Dieuwke Hupkes. 2018. Do Language Models Understand Anything? On the Ability of LSTMs to Understand Negative Polarity Items. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 222–231.

Jaap Jumelet and Dieuwke Hupkes. 2019. diagnnose: A neural net analysis library.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. In *Proceedings of NAACL-HTL 2019*, pages 11–20. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *TACL*, 4:521–535.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 4765–4774.

Rebecca Marvin and Tal Linzen. 2018. Targeted Syntactic Evaluation of Language Models. In *EMNLP*, pages 1192–1202. Association for Computational Linguistics.

Grégoire Montavon. 2019. *Gradient-Based Vs. Propagation-Based Explanations: An Axiomatic Comparison*, pages 253–265. Springer International Publishing, Cham.

W. James Murdoch, Peter J. Liu, and Bin Yu. 2018. Beyond word importance: Contextual decomposition to extract interactions from lstms. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Christopher Olah. 2015. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

Nina Poerner, Benjamin Roth, and Hinrich Schütze. 2018. Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 340–350, Stroudsburg, PA. Association for Computational Linguistics (ACL).

Lloyd S. Shapley. 1953. A value for n-person games. *Contributions to the Theory of Games*, (28):307–317.

Chandan Singh, W. James Murdoch, and Bin Yu. 2019. Hierarchical interpretations for neural network predictions. In *ICLR*.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN Language Models Learn about Filler-Gap Dependencies? *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20.

Deconstructing Supertagging into Multi-task Sequence Prediction

Zhenqi Zhu
School of Computing Science
Simon Fraser University
Burnaby, BC , Canada
zhenqiz@sfu.ca

Anoop Sarkar
School of Computing Science
Simon Fraser University
Burnaby, BC , Canada
anoop@sfu.ca

Abstract

Supertagging is a sequence prediction task where each word is assigned a piece of complex syntactic structure called a supertag. We provide a novel approach to multi-task learning for Tree Adjoining Grammar (TAG) supertagging by deconstructing these complex supertags in order to define a set of related but auxiliary sequence prediction tasks. Our multi-task prediction framework is trained over the exactly same training data used to train the original supertagger where each auxiliary task provides an alternative view on the original prediction task. Our experimental results show that our multi-task approach significantly improves TAG supertagging with a new state-of-the-art accuracy score of 91.39% on the Penn treebank supertagging dataset.

1 Introduction

A treebank for lexicalized tree-adjoining grammar (TAG) (Joshi and Schabes, 1997) consists of annotated sentences where each word is provided a complex tree structure called a supertag and the overall parse of the sentence combines these supertags into a parse tree. Supertagging is a task that learns a sequence prediction task from this annotated data and is able to then assign the most likely sequence of supertags to an input sequence of words (Bangalore and Joshi, 1999). Once the right supertag is assigned then parsing is a much easier task and may not even be needed for many applications where information about syntax is needed but a full parse is unnecessary.

Supertagging has been shown to be useful for both Tree Adjoining Grammar (TAG) (Bangalore and Joshi, 1999) and combinatory categorial grammar (CCG) (Hockenmaier and Steedman, 2007) parsing. In this paper we aim to improve the state-of-the-art for the task of learning a TAG supertagger from an annotated treebank (Kasai et al., 2018). We observe that supertag prediction does not take full advantage of the complex structural information contained within each supertag. Neural models have been used to learn embeddings over these supertags and thereby share weights among similar supertags. Friedman et al. (2017) provide tree-structured neural models over supertags which can learn interesting relationships between supertags but the approach does not lead to higher supertagging accuracy.

Our main contribution is to provide several novel ways to deconstruct supertags to create multiple alternative auxiliary tasks, which we then combine using a multi-task prediction framework and we show that this can lead to a significant improvement in supertagging accuracy.

Multi-task learning (MTL) (Caruana, 1997) learns multiple heterogenous tasks in parallel with a shared representation so that what is learned for one task can be shared for another task. In most cases the improvement is due to weight sharing between different tasks (Collobert and Weston, 2008; Luong et al., 2015). While some combinations may not provide any benefit in MTL (Bingel and Søgaard, 2017) and the improvements might be simply due to training on more data. However, MTL can be effective even when using large pretrained models (Liu et al., 2019).

Unlike most other work in multi-task learning with neural models we do not use different annotated datasets for each task. Similar to the approach to combining different representations for phrase structure parsing in (Vilares et al., 2019) we also construct multiple tasks from exactly the same training data set. Our approach is also distinct in that we take advantage of the structure of the supertags by deconstructing the tree structure implicit in each supertag.

Our experimental results show that our novel multi-task learning framework leads to a new

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 12–21
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

state-of-the-art accuracy score of 91.39% for TAG supertagging on the Penn Treebank dataset (Marcus et al., 1993; Chen et al., 2006) which is a significant improvement over the previous multi-task result for supertagging that combines supertagging with graph-based parsing (Kasai et al., 2018).

2 The Supertagging Task

Supertagging assigns complex structural descriptions to each word in the sentence. The complex structural descriptions come from grammar formalisms that are more expressive than context-free grammars for phrase structure trees or dependency trees. In Tree Adjoining Grammar (TAG), the supertags are tree fragments that can express various syntactic facts such as transitive verb, wh- extraction, relative clauses, appositive clauses, light verbs, prepositional phrase attachment and many other syntactic phenomena. In combinatory categorial grammar (CCG) the supertags are types and their type-raised variants which also capture similar syntactic phenomena as in TAG supertags. Supertagging can be viewed as "almost parsing" (Bangalore and Joshi, 1999) and can provide the benefits of syntactic parsing without a full parser.

In this paper we focus on the TAG supertagging task, however, our proposed methods can likely be used to improve CCG supertagging as well. Supertagging is a relatively simple linear time sequence prediction task similar to part of speech tagging. Supertagging can be useful in many applications such as machine translation, grammatical error detection, disfluency prediction, and many others while being a much simpler task than full parsing.

In addition, for both TAG and CCG, supertagging is an essential first step to parsing so any improvements in supertag prediction will benefit parsing as well. For all these reasons, in this paper we focus on the supertagging task. TAG and CCG can be parsed using graph-parsing methods in $O(n^3)$ but the complexity of unrestricted parsing for both formalisms is $O(n^6)$ which is prohibitive on real-world data. Neural linear-time transition based parsers are still not accurate enough to compete with the state-of-the-art supertagging models or parsers that use supertagging as the initial step (Chung et al., 2016; Kasai et al., 2018).

An example of the supertagging task for Tree Adjoining Grammars (TAGs) is shown in Fig. 1. The $\downarrow$ symbol on a leaf node represents a substitution node which can be expanded by a tree rooted in the same label, e.g. t3 rooted in NP substitutes into the NP$\downarrow$ node in t46. The $*$ symbol on the leaf node of a tree t represents an adjunction node (also called a footnode) and signifies that t can be inserted into an internal node of another tree with the same label, e.g. t103 adjoins into the AP node in t46. The $\diamond$ node is called the *head* and represents the node where the word token is inserted into the tree. The table on the right shows how many different supertags are possible for each word in the sentence.

Three factors make supertagging a challenging task for sequence prediction: much more severe token level ambiguity when compared to other like part-of-speech tagging, a large number of distinct supertag types (4727 distinct supertags in our dataset, including an unknown supertag) and a complex internal structure for each supertag.

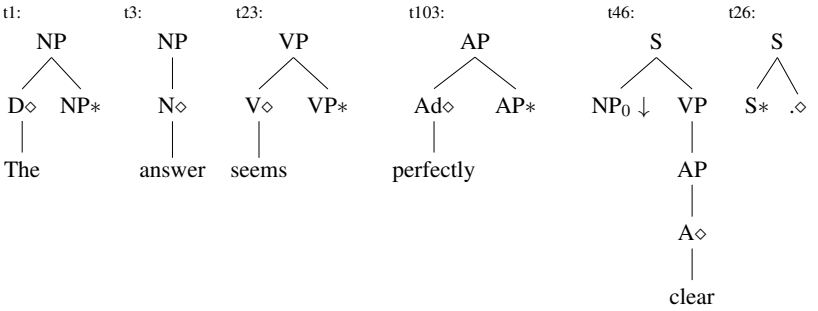
3 Baseline Supertagging Model

For our baseline supertagging model we use the state-of-the-art model that currently has the highest accuracy on the Penn treebank dataset (Kasai et al., 2018). For the supertagging model the main contribution of Kasai et al. (2018) was two-fold: the first was to add a character CNN for modeling word embeddings using subword features, and the second was to add highway connections to add more layers to a standard bidirectional LSTM. The output layer was a standard multi-layer perceptron that had a softmax output over the set of supertags. Another extension to the standard sequence prediction model in Kasai et al. (2018) was to combine supertagging with graph-based parsing.

In this paper, we focus on the supertagging model and compare only on supertagging accuracy. The neural model for supertagging that we use as a baseline uses graph-based parsing as an auxiliary task and has the current highest accuracy score on the Penn treebank (90.81%). The model has three main components: the input layer, the bidirectional LSTM component, and the output layer which computes a softmax over the set of supertags. The input to the model is a sequence of words and the output is a sequence of supertags, one per word, which makes it a standard tagging aka sequence prediction task.

3.1 Input Layer

Each word in the input sequence is converted into a word embedding in the input layer. Following



Token	#supertags
The	5
answer	14
seems	20
perfectly	5
clear	32
.	16

Figure 1: An example that explains the supertagging task for Tree Adjoining Grammars (TAGs). For the sentence "*The answer seems perfectly clear .*" the correct supertag for each word is shown above. The table on the right shows how many different supertags are possible for each word in the sentence. See Section 2 for more details on the notation used to define the supertags and how the supertags can be combined to form a parse tree.

(Kasai et al., 2018) we use two components in the word embedding:

- a 30-dimensional character level embedding vector computed using a char-CNN which captures the morphological information (Santos and Zadrozny, 2014; Chiu and Nichols, 2016; Ma and Hovy, 2016; Kasai et al., 2018). Each character is encoded as a 30-dimensional vector, and then we apply 30 convolutional filters with a window size of 5. This produces a 30-dimensional character embedding.

- a 100/200/300 size word embedding which is initialized using GloVe (Pennington et al., 2014). For words that do not appear in GloVe, we randomly initialized the word embedding.

A start of sentence token and an end of sentence token is added into the beginning and ending position of each sentence, but is not included in the computation of loss and accuracy.

Unlike (Kasai et al., 2018) we do not use predicted part of speech (POS) tags as part of the input sequence. In our experiments, the improvement was negligible and there was a significant overhead of having to do part of speech predictions at test time.

3.2 BiLSTM Layer

The core of this base model is a bidirectional recurrent neural network, in particular a Long Short-Term Memory neural network (Graves and Schmidhuber, 2005). For the hyperparameters, we use the settings in Kasai et al. (2018) in order to ensure a fair comparison.

Unlike (Kasai et al., 2018) we do not use highway connections in our model. We did experiment with the addition of highway connections but we found no improvement in accuracy over the baseline BiLSTM-only model with a significant increase in training time.

The bidirectional representation has 1024 units, a combination of the 512 forward and backward units each. Dropout layers (Gal and Ghahramani, 2016; Srivastava et al., 2014) are inserted between the input and BiLSTM layer, between BiLSTM layers, and between recurrent time steps. The dropout rate used was 0.5. We used 2-3 BiLSTM layers. Kasai et al. (2018) provide some reasons why > 3 layers do not provide any additional accuracy even with highway connections.

3.3 Output Layer

We concatenate hidden vectors from both directions of the last layer of BiLSTM and pass it into a multilayer perceptron (MLP). In practice a single layer perceptron performs just as well in this task. The number of input neurons of the single layer perceptron equals 1024 (2×512) and the output vector size equals the number of labels for each specific task: 4727 for the main supertagging task.

4 Deconstructing Supertags

The error analysis of our baseline BiLSTM model is shown in Fig. 1. We observed some consistent ways in which the baseline model confused the correct supertag with the incorrect one. We also observed that the baseline BiLSTM model can achieve over 97% 3-best accuracy on the supertagging task. This means it should be possible to boost the accuracy by rescoring the alternatives that already exist in the n-best output of the baseline supertagger. Rather than a re-ranking frame-

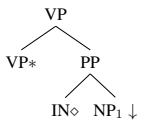
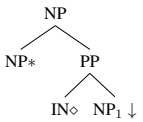
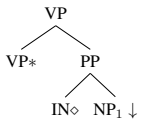
Prediction	Ground Truth	#times in dev
t2: 	t36: 	194
tCO: CO◇	t13: 	156
t4: 	t13: 	144

Table 1: The top-3 errors made by the state-of-the-art Bi-LSTM supertagger. tCO stands for a co-head in the case where a supertag has multiple heads. One example is a sentence fragment like *pull it from the marketplace* which contains a multi-word predicate *pull ... from*; *pull* is the V◇ head of tree t531 which has a IN^c node where tCO (headed by *from*) is inserted.

work we used a multi-task learning framework in order to boost the scores of correct supertags over the error-prone supertags. The auxiliary tasks we created based on our error analysis are as follows.

4.1 Auxiliary Tasks

4.1.1 HEAD

Consider the trees t2 and t36 in Table 1. t2 is headed by a noun head N and t36 is headed by an adjective A. The label of the head node is a useful auxiliary task for disambiguation. We define a function HEAD(t) to get the head node (marked by a diamond) of supertag t. There are 29 distinct HEAD labels.

4.1.2 ROOT

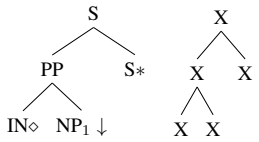
Consider the trees t4 and t13 in Table 1. t4 modifies an NP node while t13 modifies a VP node. This is a case of preposition attachment ambiguity. The label of the root node is a useful auxiliary task for disambiguation. We define a function ROOT(t) to get the root node of supertag t. There are 48 distinct ROOT labels.

4.1.3 TYPE

Consider the trees tCO and t13 in Table 1. tCO is a supertag that does not use adjunction (this type of supertag is called an initial tree). In contrast, t13 modifies an internal VP node in another supertag (this type of supertag is called an auxiliary tree). In addition a left auxiliary tree modifies from the left while a right auxiliary tree modifies from the right. To make this task more sensitive we also include the node label of the root (for initial trees) or footnode which is the node marked with $*$ (for left/right auxiliary trees). We define a function TYPE(t) to obtain the type of each supertag. There are 67 distinct types.

4.1.4 SKETCH

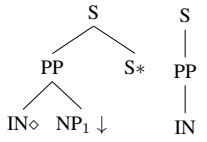
In many cases, the overall shape of the supertag is useful for disambiguation, ignoring the node labels. The following example keeps the tree structure of the supertag but removes the node labels:



Tree sketches help disambiguation (see t81 in Table 5). We define a function SKETCH(t) that returns the sketch. There are 602 distinct supertag sketches.

4.1.5 SPINE

The spine of a supertag is the path from the root node to the head node (marked by ◇). The following example keeps only the path from root to head and produces a spine supertag:



Spine supertags are helpful for disambiguation as well (see t132 in Table 5). We use a function SPINE(t) to return the spine of supertag t. There are 1372 distinct supertag spines.

4.2 Multi-task Framework

Unlike most other work in multi-task learning with neural models we do not use different datasets for each task. We use exactly the same training data set but we construct multiple tasks with alternate output labels by automatically deconstructing the supertags (the output labels in the original task). These alternate output labels are easier to predict than the full set of supertags, and these new output labels are related to the original supertag in a linguistically relevant way. As a result, we train on the same training set but with alternate output labels, each forming a different task. We then combine these multiple tasks in order to improve the performance in the original supertagging task.

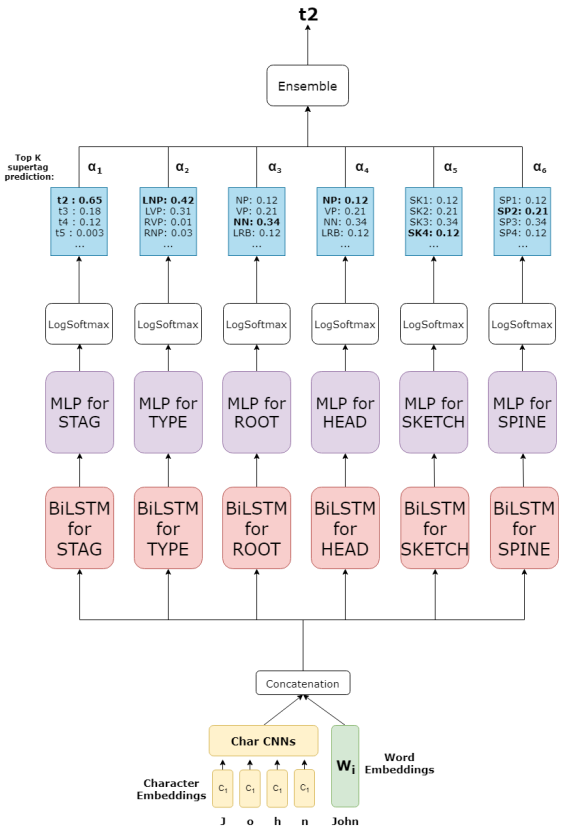


Figure 2: The prediction procedure of combining models trained on separate tasks

The usual criticism of a fair comparison between multi-task and single-task learning is that the multi-task setting simply uses more labeled data instances (typically with different data sources) and as a result a fair comparison between a multi-task and a single-task setting should involve large pre-trained models trained using a language modelling objective (such as ELMO (Peters et al., 2018) or BERT (Devlin et al., 2018)). In our case, because we re-use the same training set for multi-task learning, we have made sure our experimental settings exactly match the previous best state-of-the-art method for supertagging (Kasai et al., 2018) and we use the same pre-trained word embeddings to ensure a fair comparison.

We train six different neural sequence prediction models **independently** on the supertagging task, root node prediction (ROOT), head node prediction (HEAD), tree type prediction (TYPE), tree sketch prediction (SKETCH) and tree spine prediction (SPINE) tasks. For each task, we use the state-of-the-art baseline supertagging model as defined in Section 3. The only change is that the output size for softmax is changed to reflect the number of output labels in each task. We obtain very high accuracies for each of the tasks. For example, on the dev set we obtain the following accuracies: ROOT = 97.04%, HEAD = 93.37%, TYPE = 93.14%, SKETCH = 93.74% and SPINE = 91.00%.

We train the model, including the word embedding (which is initialized using a pre-trained embedding) and character-level CNNs by optimizing the negative log-likelihood of the predicted sequences of output labels. The output labels for each task is different: supertag, root node, head node, tree type, sketch, spine. Training is done using minibatches. The main hyperparameters are as follows: we use the ADAM optimizer with a batch size of 100 and learning rate $\ell = 0.001$ (Kingma and Ba, 2015). After every training epoch, we evaluate the model on the dev set, if the accuracy on dev set has not been improved for five consecutive epochs, training stops. The maximum number of epochs is 70. After obtaining the best model trained with $\ell = 0.001$, we further fine-tune the best model using $\ell = 0.0001$ for at most 10 more epochs. By conducting this step, we have seen 0.1% to 0.2% accuracy improvement depending on the task.

After obtaining the best trained model on each of the multiple tasks we combine the multiple tasks together in order to create a decoder for the supertagging task.

We first run the baseline supertagger to obtain the distribution P_{STAG} and using this distribution we select the top-K output supertags for each word in each sentence in the dev or test data. We experiment with different values of K but we know that even K=3 gives 97% accuracy for the supertagging task. For each dev or test sentence we also compute the output softmax distributions for each task, P_{HEAD}, P_{ROOT}, P_{TYPE}, P_{SKETCH}, P_{SPINE}. Each of these probabilities are defined as a sequence prediction task over the auxiliary tasks using the functions defined in Section 4.1.

$$
\begin{aligned}
P_{\text{HEAD}}(t) &= P(\text{HEAD}(t)) \\
P_{\text{ROOT}}(t) &= P(\text{ROOT}(t)) \\
P_{\text{TYPE}}(t) &= P(\text{TYPE}(t)) \\
P_{\text{SKETCH}}(t) &= P(\text{SKETCH}(t)) \\
P_{\text{SPINE}}(t) &= P(\text{SPINE}(t))
\end{aligned}
$$

We compute the argmax sequence of supertags $t_1^*, t_2^*, \ldots, t_T^*$ by scoring each supertag t_i^* individ-

ually from the top-K list by combining the probabilities from the different tasks as follows:

$$t_i^* = \arg\max_{t_i \in S} \quad (1)$$

$$\alpha_1 P_{\text{STAG}}(t_i) + \alpha_2 P_{\text{HEAD}}(t_i) \quad (2)$$

$$+\alpha_3 P_{\text{ROOT}}(t_i) + \alpha_4 P_{\text{TYPE}}(t_i) \quad (3)$$

$$+\alpha_5 P_{\text{SKETCH}}(t_i) + \alpha_6 P_{\text{SPINE}}(t_i) \quad (4)$$

S is the top-K set of supertags for each word in the input sequence. The hyperparameters α_i can be tuned. However we found in our experiments that the results were not very sensitive to the values, and the uniform distribution over all the tasks performed the best. The model and decoding step for our multi-task model is shown in Fig. 2. We also experiment with a commonly used multi-task model where some or all of the components are shared between the different (unlike our approach)..

5 Dataset

We use the dataset that has been widely used by previous work in supertagging and TAG parsing (Bangalore et al., 2009; Chung et al., 2016; Friedman et al., 2017; Kasai et al., 2017, 2018). We use the grammar and the TAG-annotated WSJ Penn Tree Bank extracted by Chen et al. (2006). As in previous work, we use Sections 01-22 as the training set, Section 00 as the dev set, and Section 23 as the test set. The training, dev, and test sets comprise 39832, 1921, and 2415 sentences; 950028, 46451, 56683 tokens, respectively.

The TAG-annotated version of Penn treebank (Chen and Shankar, 2001) includes 4727 distinct supertags (including an unknown supertag) and the grammar file of all supertags is downloaded from `http://mica.lif.univ-mrs.fr/`. There are 69 auxiliary tree TYPEs, 40 distinct types of ROOT node and 30 different types of HEAD node, 602 tree SKETCHes and 1372 tree SPINEs.

6 Results and Discussion

For our experiments, we implemented all of the models we discussed above in PyTorch (Paszke et al., 2017). We have various hyperparameters and Table 2 shows the results obtained from the different model configurations which were described in Section 3. The table also includes the results from the multi-task model and decoder described in Section 4. We experiment with pretrained GloVe word embeddings of three different sizes: 100, 200 and 300.

With our multi-task approach, all base models gain significant improvements compared to a single supertagging base model between 0.4% to 0.65%. We also varied the parameter K which picks the top-K supertags from the baseline model for use with the multi-task model. Table 3 that increasing K helps up to a point. After K=10 there is no further improvement.

We obtain a new state-of-the-art result of 91.39% which is significantly better than the 90.81% result which combines supertagging with the parsing task and so is using more labeled training information used by our supertagger models.

Table 4 shows the result of task ablation for each task. We can see that adding a new task always improves the results. The best result is obtained by using all five auxiliary tasks.

We computed a significance score on the accuracy of our best model BiLSTM3+CNN+GloVe200 with and without multi-task learning. On the dev set, using McNemar's significance test we found that the multi-task model is significantly better than the baseline model with a p-value of 0.0062; on the test set, the p-value is 0.0064.

We evaluated our own implementation of the baseline BiLSTM-only model and even with highway connections we only obtained 89.25% on the dev set compared to the built-in BiLSTM implementation in Pytorch (without highway connections) which obtains 89.94%.

6.1 Task Contribution

Table 5 shows some examples about how each of auxiliary tasks can help in the correction of supertag prediction. Examples of each task are selected if a considerable number of predictions of each example are corrected after applying the multi-task model.

While the multi-task model can correct many wrong predictions made by the baseline model, the multi-task model may also override some correct predictions.

The first row is an example of the prediction of head node that helps differentiate two similar supertags, t2 and t36. In the dev set, there are 24 words of which ground truth supertags are t2, wrongly predicted as t36 by a single base model; 25 words of which ground truth supertags are t36, wrongly predicted as t2. All of those words are

Model	Multi-task	Dev	Test
BiLSTM3+HW+CNN+POS+GloVe100 (Kasai et al., 2018)	-	90.45	90.81
BiLSTM2+GloVe100	No	89.11	-
	Yes	89.67	-
BiLSTM2+CNN+GloVe100	No	89.45	-
	Yes	90.12	-
BiLSTM3+GloVe100	No	89.41	-
	Yes	90.02	-
BiLSTM3+CNN+GloVe100	No	89.83	-
	Yes	90.41	-
BiLSTM3+CNN+GloVe200	No	89.94	-
	Yes	**90.55**	**91.37**
BiLSTM3+CNN+GloVe300	No	89.91	-
	Yes	90.45	-
Shared BiLSTM layer (BiLSTM3+CNN+Glove200)	No	90.11	90.83
	Yes	90.11	90.83

Table 2: Supertagging task results. The number after BiLSTM represents the number of BiLSTM layers; CNN refers to the word embedding model using character-level CNN; the number immediately after GloVe represents the dimension of pre-trained GloVe word vectors. HW in Kasai et al. (2018) refers to highway connections, and POS refers to the use of predicted part-of-speech tags as inputs. We do not use HW or POS in our models as they do not provide any benefit.

Top-K	Dev	Test
Top-3	90.55	91.37
Top-5	90.58	91.38
Top-10	**90.58**	**91.39**
Top-20	90.58	91.39

Table 3: Change in accuracy as K is increased when choosing Top-K supertags for rescoring. The model used is BiLSTM+CNN+GloVe200.

correctly predicted by the multi-task model. The ROOT, TYPE, SKETCH and SPINE are all the same for t2 and t36, the only difference is the HEAD value, **N** for t2 and **A** for t36. The model for the HEAD task correctly predicts the head node of those words which is further improved using our multi-task approach.

The second row demonstrates how the tree sketch can help discriminate supertags. t81 and t27 have exactly the same ROOT, HEAD, SPINE (S-VP-V) and TYPE (Init), the only difference between these two supertags is the tree structure.

The third to fifth rows are examples of the effect of multiple auxiliary tasks in getting the prediction right. The third row is an example of the prediction of TYPE and SKETCH that can help differentiate supertags. The TYPE of t3 is **Init**, while t38 has TYPE **Left+NP**. They also have different tree sketches. There are 11 words of which supertags are wrongly predicted as t3 by a single supertagging model, but correctly predicted as t38 by the multi-task model; also, 3 words of which supertags are wrongly predicted as t38 by a single supertagging model, but correctly predicted as t3 by the multi-task model.

The forth row is an example of how the prediction of the ROOT can help differentiate supertags. The ROOT of t3 is **NP**, while t18 has ROOT **N** (N is also its head node). For the last row, t132 and t20 have the same root node(S), head node(Punct) and tree type (Right+S) but they are different in the tree spine (**S-Punct** for t20 and **S-PRN-Punct** for t132) and SKETCH. The joint effort of various models plays a significant role in getting the prediction right.

7 Related Work

Bangalore et al. (2009) and Chung et al. (2016) trained a feature based classification model for TAG supertags, that extract features using lexical, part-of-speech attributes from the left and right context in a 6-word window and the lexical, orthographic (e.g. capitalization, prefix, suffix, digit) and part-of-speech attributes of the word being supertagged. Neural network based supertagging models in TAG (Kasai et al., 2018) and CCG (Xu

Multi-task setting	Dev	Test
None	89.94	90.73
HEAD	90.00	90.79
ROOT	90.06	90.91
TYPE	90.15	91.07
SKETCH	90.25	90.99
SPINE	90.22	91.08
HEAD+ROOT	90.15	90.94
TYPE+HEAD+ROOT	90.27	91.10
TYPE+HEAD+ROOT+SKETCH	90.48	91.27
TYPE+HEAD+ROOT+SKETCH+SPINE	**90.55**	**91.37**

Table 4: Result of different multi-task combinations. The base model is BiLSTM+CNN+GloVe200.

Ground truth	Baseline	Multi-Task	Most Helpful Task
t2: NP → N◇ NP∗	t36: NP → A◇ NP∗	t2: NP → N◇ NP∗	HEAD
t81: S → NP_0↓ VP → V◇	t27: S → NP_0↓ VP → V◇ NP_1↓	t81: S → NP_0↓ VP → V◇	SKETCH
t3: NP → N◇	t38: NP → NP∗ NP → N◇	t3: NP → N◇	TYPE, SKETCH
t3: NP → N◇	t18: N◇	t3: NP → N◇	ROOT, SKETCH
t132: S → PRN S∗ → Punct◇	t20: S → Punct◇ S∗	t132: S → PRN S∗ → Punct◇	SPINE, SKETCH

Table 5: Some examples of how the deconstructing of base models correct the prediction made by the supertagging model.

et al., 2015; Lewis et al., 2016; Xu, 2016; Vaswani et al., 2016) have shown substantial improvement in performance, but the supertagging models are all quite similar as they all use a bi-directional RNN feeding into a prediction layer. Structural features of supertags are heavily used in pre-neural statistical parsing methods (Bangalore et al., 2009) and proved to be useful. The use of supertag structure was explored in (Friedman et al., 2017) where they adopt grammar features into a tree-structured neural model over the supertags but this model was unable to beat the state-of-the-art. (Kasai et al., 2018) combines supertagging with parsing which does provide state-of-the-art accuracy but at the expense of computational complexity.

Kasai et al. (2017) extends the BiLSTM model with predicted part-of-speech tags and suffix embeddings as inputs, then Kasai et al. (2018) further extends the BiLSTM model with highway connection as well as character CNN as input, and jointly train the supertagging model with parsing model and this work had the state-of-the-art accuracy before our paper on the Penn treebank dataset. Friedman et al. (2017) investigated a recursive tree-based vector representation of TAG supertags, but while their model can learn useful facts about supertags, about how one can be related to another, there was no performance improvement as a result of their model on the supertagging task. Xu et al. (2015) uses RNN for the CCG supertagging task, Lewis et al. (2016) adopted the LSTM structure into this task, while Vaswani et al. (2016) also introduced another variation of Bi-LSTM into this task. Xu (2016) then proposed an attention-based Bi-LSTM supertagging model.

8 Conclusion

In this paper we have introduced a novel multi-task framework for the TAG supertagging task. The approach involved a novel multi-task learning framework which led to a new state-of-the-art accuracy score of 91.39% for TAG supertagging on the Penn treebank dataset.

Our multi-task prediction framework is trained over the exactly same training data used to train the original supertagger where each auxiliary task provides an alternative view on the original pre-

diction task.

In the future we would like to explore further tasks to integrate into our multi-task sequence prediction framework. We also believe that the idea of our multi-task framework can be applied into similar tasks such as CCG supertagging task of which the labels themselves contains the latent information. We would also like to investigate how to semi-automatically generate new tasks which can be of further help in the multi-task setting.

Acknowledgments

We would like to thank Logan Born and the anonymous reviewers for their helpful comments. The research was also partially supported by the Natural Sciences and Engineering Research Council of Canada grants NSERC RGPIN-2018-06437 and RGPAS-2018-522574 and a Department of National Defence (DND) and NSERC grant DGDND-2018-00025 to the second author.

References

Srinivas Bangalore, Pierre Boullier, Alexis Nasr, Owen Rambow, and Benoît Sagot. 2009. Mica: A probabilistic dependency parser based on tree insertion grammars (application note). In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 185–188.

Srinivas Bangalore and Aravind K Joshi. 1999. Supertagging: An approach to almost parsing. *Computational linguistics*, 25(2):237–265.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain. Association for Computational Linguistics.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

John Chen, Srinivas Bangalore, and K Vijay-Shankar. 2006. Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering*, 12(3):251–299.

John Chen and Vijay Shankar. 2001. *Towards efficient statistical parsing using lexicalized grammatical information*. Ph.D. thesis, Citeseer.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Wonchang Chung, Siddhesh Suhas Mhatre, Alexis Nasr, Owen Rambow, and Srinivas Bangalore. 2016. Revisiting supertagging and parsing: How to use supertags in transition-based parsing. In *12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 12)*, pages 85–92.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dan Friedman, Jungo Kasai, R Thomas McCoy, Robert Frank, Forrest Davis, and Owen Rambow. 2017. Linguistically rich vector representations of supertags for tag parsing. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 122–131.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: a corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

Aravind K Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer.

Jungo Kasai, Robert Frank, R Thomas McCoy, Owen Rambow, and Alexis Nasr. 2017. Tag parsing with neural networks and vector representations of supertags. In *Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722.

Jungo Kasai, Robert Frank, Pauli Xu, William Merrill, and Owen Rambow. 2018. End-to-end graph-based TAG parsing with neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1181–1194.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. Lstm ccg parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Ashish Vaswani, Yonatan Bisk, Kenji Sagae, and Ryan Musa. 2016. Supertagging with lstms. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 232–237.

David Vilares, Mostafa Abdou, and Anders Søgaard. 2019. Better, faster, stronger sequence tagging constituent parsers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3372–3383, Minneapolis, Minnesota. Association for Computational Linguistics.

Wenduan Xu. 2016. Lstm shift-reduce ccg parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1764.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. Ccg supertagging with a recurrent neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 250–255.

Multilingual model using cross-task embedding projection

Jin Sakuma
The University of Tokyo
jsakuma@tkl.iis.u-tokyo.ac.jp

Naoki Yoshinaga
Institute of Industrial Science,
the University of Tokyo
ynaga@iis.u-tokyo.ac.jp

Abstract

We present a method for applying a neural network trained on one (resource-rich) language for a given task to other (resource-poor) languages. We accomplish this by inducing a mapping from pre-trained cross-lingual word embeddings to the embedding layer of the neural network trained on the resource-rich language. To perform element-wise cross-task embedding projection, we invent locally linear mapping which assumes and preserves the local topology across the semantic spaces before and after the projection. Experimental results on topic classification task and sentiment analysis task showed that the fully task-specific multilingual model obtained using our method outperformed the existing multilingual models with embedding layers fixed to pre-trained cross-lingual word embeddings.[1]

1 Introduction

Deep neural networks have improved the accuracy of various natural language processing (NLP) tasks by performing representation learning with massive annotated datasets. However, the annotations in NLP depend on the target language as well as the task, and it is unrealistic to prepare such extensive annotated datasets for every pair of language and task. As a result, we can only obtain an accurate model for a few resource-rich languages such as English.

To overcome this problem, researchers have attempted to make models trained with massive annotated datasets in a resource-rich language (hereafter, *source language*) applicable to a resource-poor language (*target language*) that have no annotated datasets (Ruder et al., 2019) (§ 2). These methods utilize language-universal word representations, namely cross-lingual word embeddings, to absorb the differences among languages in the vocabularies of neural network models; specifically, these *multilingual models* are trained with embedding layers fixed to pre-trained cross-lingual word embeddings. However, because those embedding layers are not optimized for the target task, the resulting model cannot exploit the true potential of representation learning, as demonstrated by Kim (2014) and our experimental results (§ 5.1).

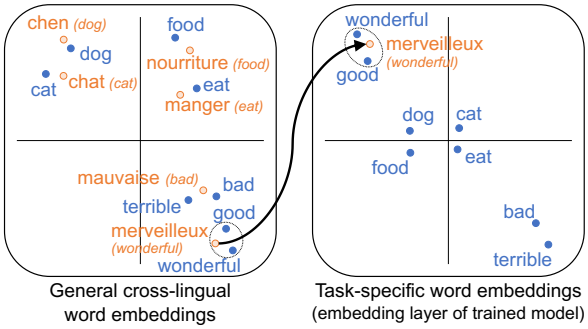


Figure 1: Locally linear mapping for sentiment analysis task. The relationship between "merveilleux (wonderful)" and its neighboring English words, "wonderful" and "good," are preserved after projection.

We propose methods of projecting pre-trained cross-lingual word embeddings to word embeddings of a *fully* task-specific neural network all of whose parameters are optimized to the training data in a source language, to realize *fully task-specific multilingual model* (§ 3). In addition to naive linear projection, we present an element-wise projection method inspired by locally linear embeddings used for dimension reduction (Roweis and Saul, 2000). This method is built on the assumption that local topology is preserved between the semantic spaces of word embeddings in two NLP tasks; that is, *adequately close* words in pre-trained cross-lingual word embeddings will have similar representation even in task-specific semantic space (Figure 1). We first represent the

[1] All the code is available at: https://github.com/jyori112/task-spec

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 22–32
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

general cross-lingual word embedding of a word in the target language by weighted linear combinations of general cross-lingual word embeddings of k neighboring words in the source language. We then use the weights to compute a task-specific word embedding of the target word as a linear combination of task-specific word embeddings of the k neighboring source words (§ 3.2).

We evaluate our method on topic classification and sentiment analysis tasks (§ 4). We first obtain a task-specific neural network using annotated corpora in the source language (English) and then induce task-specific cross-lingual word embeddings for the target languages to apply the accurate task-specific neural network to those languages. Experimental results demonstrate that our method has improved the classification accuracy of the multilingual model (Duong et al., 2017) in most of the task-language pairs (§ 5).

Our contributions are as follows:

- We established **a method of obtaining fully task-specific multilingual models** by learning a cross-task embedding projection (§ 3).
- Our **cross-task projection is simple and has an analytical solution** with one hyperparameter; the solution is a global optima (§ 3.2).
- We **confirmed the limitation of the traditional multilingual model** with embedding layers fixed to pre-trained cross-lingual word embeddings (§ 5.1).
- We **showed the effectiveness of our method** over the existing models (§ 5.2).

2 Related work

Lack of resources in resource-poor languages has been a deeply rooted problem in NLP, and there have been many pieces of researches contributed to mitigating this problem by transferring models across languages.

Multilingual models using parallel corpora An intuitive approach to realize the cross-lingual transfer of a model is to utilize machine translation by either translating the training set or the model input (Wan, 2009). Instead of translating, Meng et al. (2012) leverage a parallel corpus of the source and target languages to obtain cross-lingual mixture model to bridge the language gap. Xu and Wan (2017) also utilize parallel corpus with word alignment to train a multilingual model for sentiment analysis task. While some of these methods do not rely on an annotated corpus in the target language, they heavily rely on cross-lingual resources such as parallel corpora, and thus, are not applicable to the resource-poor languages.

Multilingual models with cross-lingual word embeddings Another method to obtain multilingual models is to fix the embedding layer of a neural network to pre-trained cross-lingual word embeddings. Many existing pieces of researches implemented this for various tasks in unsupervised senario (Duong et al., 2017; Can et al., 2018) where no annotated corpus is available in the target language as ours and supervised scenario (Pappas and Popescu-Belis, 2017; Upadhyay et al., 2018) where a small annotated corpus is available in the target language. Another study enhanced this method by employing language-adversarial networks (Chen et al., 2018). These methods do not induce task-specific word embeddings, thereby failing to exert true potential of neural networks, as we confirm in § 5.

Multilingual models with character embeddings Several studies utilize character level embeddings shared across languages to obtain multilingual models (Kim et al., 2017; Yang et al., 2017). An obvious weak point of these methods is that they do not apply to distant language pairs with different alphabets. In contrast, our method only relies on cross-lingual word embeddings which are obtainable regardless of the alphabets of the languages (Artetxe et al., 2018).

Task-specific word embeddings Few efforts have been previously made to obtain cross-lingual task-specific word embeddings. Gouws and Søgaard (2015) obtain task-specific cross-lingual word embeddings by constructing a task-specific bilingual dictionary, which defines "equivalent classes" designed for the given task instead of equivalent semantics. Although they successfully obtained task-specific cross-lingual word embeddings for POS tagging and supersense tagging tasks, the open problems are how to define a task-specific bilingual dictionary for many of other tasks, and cost of developing such resources.

Feng and Wan (2019) exploit multi-task learning to induce cross-lingual task-specific word embeddings for sentiment analysis task. This method is tailored for the sentiment analysis task and thus, not applicable to other tasks.

3 Fully task-specific multilingual model

Our method first learns a neural network model by optimizing to the annotated corpus in the source language. It then induces a projection from the semantic space of general cross-lingual word embeddings to the semantic space of the optimized embedding layer, to make the model applicable to languages other than the source language.

3.1 Framework

The entire framework of obtaining a fully task-specific multilingual model is as follows:

Step 1 (train task-specific neural network) First, we train a neural network $f(\cdot; X^{\text{spec}}, \theta)$ on an annotated corpus in the source language. The embedding layer, X^{spec}, of the resulting neural network consists of task-specific word embeddings of the source language, and θ is the collection of the other parameters. At this point, this neural network is only applicable to the source language since we do not have task-specific word embeddings Y^{spec} of the target language in the same semantic space as X^{spec}.

Step 2 (induce cross-lingual word embeddings) Next, we obtain general cross-lingual word embeddings $\{X^{\text{gen}}, Y^{\text{gen}}\}$ in the same semantic space from raw monolingual corpora where X^{gen} and Y^{gen} are cross-lingual word embeddings of the source and target languages, respectively. Without loss of generality, we assume that X^{gen} and X^{spec} are aligned so that X_i^{gen} and X_i^{spec} represent the same word. We utilize unsupervised cross-lingual word embeddings such as (Artetxe et al., 2018) that do not require any cross-lingual resources to maximize the applicability of our approach.

Step 3 (learn cross-task embedding projection) Then, we induce a cross-task projection ϕ that computes task-specific word embeddings of the target language Y^{spec} from the general cross-lingual word embeddings $\{X^{\text{gen}}, Y^{\text{gen}}\}$ obtained in Step 2 to the task-specific word embeddings of the source language X^{spec} obtained in Step 1. We explain the details of this core part in § 3.2.

Step 4 (obtain task-specific multilingual model) Finally, we replace embedding layer X^{spec} of the neural network $f(\cdot; X^{\text{spec}}, \theta)$ trained in Step 1 with Y^{spec} induced in Step 3 to obtain a task-specific neural network $f(\cdot; Y^{\text{spec}}, \theta)$ applicable to the target language.

3.2 Cross-task embedding projection

Here, we explain the detailed construction of our cross-task projection ϕ for cross-lingual word embeddings used in Step 3 in § 3.1. Given general cross-lingual word embeddings, X^{gen} and Y^{gen}, of the source and target languages and task-specific word embeddings X^{spec} of the source language, we compute task-specific word embeddings Y^{spec} of the target language in the same semantic space with X^{spec}. In what follows, we propose two simple methods to obtain such projection: a linear projection and a locally linear mapping.

Linear projection

One naive approach is to regard general and task-specific word embeddings as embeddings of two distinct languages and to exploit a mapping method developed for cross-lingual word embeddings (Mikolov et al., 2013).[2] Concretely, we train a transformation matrix W that maps general word embeddings Y^{gen} to task-specific word embeddings Y^{spec} by minimizing

$$\hat{W} = \arg\min_{W} \sum_{i=1}^{|V_X|} \left\| W X_i^{\text{gen}} - X_i^{\text{spec}} \right\|^2 \quad (1)$$

where $|V_X|$ is the vocabulary size of the source langauge. Then, we compute the task-specific word embeddings of the target language, $\hat{Y}^{\text{spec}}$;

$$\hat{Y}_i^{\text{spec}} = W Y_i^{\text{gen}}. \quad (2)$$

Locally linear mapping

A possible limitation of the above linear projection method is the lack of representation power. Due to the difference of topologies between the general and task-specific semantic spaces, our experimental results indicate that it fails to obtain precise cross-task embedding projection (§ 5).

Therefore, we introduce an element-wise mapping method inspired by locally linear embeddings (Roweis and Saul, 2000), a dimension reduction technique. Our method assumes that the local topology among nearest neighbors will be consistent between two NLP tasks (here, language modeling and the target task). In other words, synonyms will have a similar role across NLP tasks.

We build the cross-task projection as follows. First, for each word i in the target language, we

[2] Although orthogonal mapping (Xing et al., 2015) is reported to perform better for inducing cross-lingual word embeddings, it performed worse for our purpose in preliminary experiments probably due to the strong constraint.

take k nearest neighbors (words) in the source language, $\mathcal{N}_i^{\text{gen}}$, in the semantic space of the general cross-lingual word embeddings where k is a hyperparameter, and the cosine similarity is the metric. We next obtain the reconstruction weights, $\hat{\alpha}_{ij} \in \mathbb{R}$, that restore Y_i^{gen} as a linear combination of $X_j^{\text{gen}} \in \mathcal{N}_i^{\text{gen}}$ by optimizing

$$\hat{\boldsymbol{\alpha}}_i = \arg\min_{\boldsymbol{\alpha}_i} \left\| Y_i^{\text{gen}} - \sum_{j \in \mathcal{N}_i^{\text{gen}}} \alpha_{ij} X_j^{\text{gen}} \right\|^2 \quad (3)$$

with constraint of $\sum_j \alpha_{ij} = 1$. The solution to this optimization problem can be analytically given by using the method of Lagrange multipliers as:

$$\hat{\alpha}_{ij} = \frac{\sum_l (C_i^{-1})_{jl}}{\sum_j \sum_l (C_i^{-1})_{jl}} \quad (4)$$

where

$$C_{ijl} = \left(Y_i^{\text{gen}} - X_j^{\text{gen}}\right) \cdot \left(Y_i^{\text{gen}} - X_l^{\text{gen}}\right) \quad (5)$$

(see Appendix A for the detailed derivation). We can thereby find the global optima by this analytical solution with simple computation.

We then compute Y_i^{spec} using $\hat{\boldsymbol{\alpha}}_i$ by

$$Y_i^{\text{spec}} = \sum_{j \in \mathcal{N}_i^{\text{gen}}} \hat{\alpha}_{ij} X_j^{\text{spec}}, \quad (6)$$

assuming that the local topology among $\mathcal{N}_i^{\text{gen}}$ is preserved before and after the projection. The resulting Y^{spec} is in the same semantic space with X^{spec}. Setting a large $k = |\mathcal{N}_i^{\text{gen}}|$ in the projection, we can handle words in the target language that have no direct translations in the source language (*e.g.*, *amiga*, female friend in Spanish).

Hyperparameter search In general, we choose a hyperparameter that performs best on development data in the target task and language. However, since we assume that no annotated data is available in the target language, we cannot exploit development data in the target language.

To address this issue, we apply our cross-task projection to the *source* language with various hyperparameter k; namely, represent X_i^{gen} considering k nearest neighbors $X_j^{\text{gen}} (j \neq i)$. We then choose k with the best model performance with the resulting embeddings on the development data of the target task in the source language. In § 5.2, we report results with this language-universal, yet the task-specific method of tuning. We also report results of a language- and task-specific tuning method assuming a minimal development data in the target language in addition to a naive method of fixing $k = 1$, which is equivalent to the word-by-word translation. Furthermore, we investigate the effect of value k in details in § 5.3.

Language	train	dev.	test
English (en)	653,762	10,000	10,000
Danish (da)	6,633	1000	1000
German (de)	84,550	1000	1000
Spanish (es)	12,997	1000	1000
French (fr)	69,292	1000	1000
Italian (it)	19,594	1000	1000
Dutch (nl)	590	100	1000
Portuguese (pt)	4,263	1000	1000
Swedish (sv)	8,383	1000	1000

Table 1: Number of examples for topic classification.

4 Experimental setup

We conduct a series of experiments to evaluate our fully task-specific multilingual models (§ 3) obtained by our cross-task projections of cross-lingual word embeddings (§ 3.2). Our method is language- and task-independent and is applicable to various tasks where existing multilingual models are applicable. We adopted a topic classification task and a sentiment analysis task as the target tasks for evaluation in various languages.

Topic classification is the task of predicting the topic of a given document. For this task, we use English (en) as the source language, and Spanish (es), German (de), Danish (da), French (fr), Italian (it), Dutch (nl), Portuguese (pt), and Swedish (sv) as the target languages. We use the RCV1/RCV2 dataset (Lewis et al., 2004) for this task, following Duong et al. (2017). This dataset contains news articles in various languages with labels of four categories: Corporate/Industrial, Economics, Government/Social, and Markets.

For English dataset, we randomly chose 10,000 examples as test data, another 10,000 examples as development data, and the rest as training data. For the other languages, we randomly selected 1000 examples as test data, and another 1000 examples (for Danish, 100 examples) as development data, and the rest as training data. Among the development data, we randomly chose 100 samples as the development data for an alternative, language-specific tuning of k (§ 3.2). The summary of the resulting dataset is shown in Table 1.

Sentiment analysis is a task of predicting a polarity label of the writer's attitude for a given text. We design this task to be a three-class classification of positive, negative, and neutral labels. We use datasets from two domains of restaurant review and product review to conduct this experiment. In both domains, we consider the most resource-rich language, English (en), as the source language and other languages (Spanish (es), Dutch (nl), and Turkish (tr) for restaurant review domain, and German (de), French (fr), and Japanese (ja) for product review domain) as the target languages.

To train models in restaurant review domain, we use Yelp Review dataset[3] which consists of a set of restaurant reviews with numerical ratings in the range of 1-5 given by the reviewers. We label the reviews with ratings of 1 or 2 as negative, those with ratings of 4 or 5 as positive, and the rest with ratings of 3 as neutral. Then, we randomly chose 100,000 examples as test data, another 100,000 examples as development data, and the rest as training data. For evaluation in the target languages, we use a subset of ABSA dataset (Pontiki et al., 2016), which consists of restaurant reviews in English, Spanish, Dutch, and Turkish with annotation of a polarity label of positive, negative, or neutral to each sentence. For each language, we randomly chose 100 sentences as development data for the alternative, language-specific tuning of k (§ 3.2) and the rest as test data.

For experiments in the product review domain, we use Amazon Multilingual Review dataset[4] which consists of a set of product reviews in English, German, French, Japanese with numerical ratings given in the same manner as the Yelp Review dataset. We label the reviews in the same manner as the Yelp Review dataset. For English dataset, we randomly sample 100,000 examples as development data, other 100,000 examples as test data, and the remaining 6,731,166 examples as training data. For the other languages, we randomly chose 10,000 examples as development data, another 10,000 examples as test data, and the rest as training data. Among the development data, we randomly chose 100 examples as development data for the alternative, language-specific tuning of k. The summary of the resulting datasets is shown in Table 2.

[3] https://www.yelp.com/dataset

[4] https://s3.amazonaws.com/amazon-reviews-pds/readme.html

Dataset	Language	train	dev.	test
Yelp	English (en)	5,796,996	100,000	100,000
ABSA	English (en)	-	100	1462
	Spanish (es)	-	100	1237
	Dutch (nl)	-	100	1125
	Turkish (tr)	-	100	855
Amazon	English (en)	6,731,166	100,000	100,000
	German (de)	659,121	10,000	10,000
	French (fr)	234,080	10,000	10,000
	Japanese (ja)	242,431	10,000	10,000

Table 2: Number of examples for sentiment analysis.

General cross-lingual word embeddings were obtained using a state-of-the-art unsupervised method with self-learning framework (Artetxe et al., 2018).[5] This method takes monolingual word embeddings of two languages and learns a mapping between them to obtain cross-lingual word embeddings. For monolingual word embeddings, we used pre-trained word embeddings available online (Grave et al., 2018).[6] They are word embeddings with 300 dimensions obtained by applying subword-information skip-gram (Bojanowski et al., 2017) to the Wikipedia corpus.

Preprocessing We use the tokenizer of Europarl tools[7] to tokenize all datasets except for Japanese. For Japanese, we use MeCab v0.996[8] with IPA dictionary v2.7.0. After tokenization, the tokens are lowercased to match vocabularies of the pre-trained word embeddings.

Models To evaluate the impact of our task-specific word embeddings on multilingual models and to compare the two methods for the cross-task embeddings projections we proposed in § 3, we compare the following five models.

CLWE fixed trains a bag-of-embeddings model in the target language with its embedding layers fixed to the pre-trained cross-lingual word embedding. The model takes the dimension-wise average of all embeddings of input tokens into a feedforward neural network with one hidden layer. This model is analogous to (Duong et al., 2017) except that they use the summation weighted by $\texttt{tf} \cdot \texttt{idf}$.

[5] https://github.com/artetxem/vecmap

[6] https://fasttext.cc/docs/en/crawl-vectors.html

[7] http://www.statmt.org/europarl/

[8] https://taku910.github.io/mecab/

Method	en-da	en-de	en-es	en-fr	en-it	en-nl	en-pt	en-sv
CLWE fixed	0.621	0.813	0.363	0.772	0.535	0.791	0.524	0.816
CLWE fixed + NNmap	0.593	0.843	0.448	0.815	0.583	0.794	0.554	0.503
CLWE opt (LP)	0.599	0.617	0.117	0.670	0.197	0.627	0.185	0.206
CLWE opt (LLM)								
$k = 1$	**0.694**	**0.848**	**0.764**	0.879	0.578	0.815	**0.584**	0.805
k tuned to task	0.672	0.809	0.705	**0.885**	**0.623**	0.814	0.580	**0.831**
k tuned to task/language	0.687	0.833	**0.764**	0.879	0.615	**0.837**	0.572	0.830
Monolingual	0.968	0.984	0.975	0.980	0.932	0.950	0.948	0.970

Table 3: Classification accuracy of topic classification task in cross-lingual settings. The underlined values indicate that, among the three trials, the worst model of **CLWE opt (LLM)** outperforms the best model of **CLWE fixed**.

	Amazon			Yelp - ABSA		
Method	**en-de**	**en-fr**	**en-ja**	**en-es**	**en-nl**	**en-tr**
CLWE fixed	0.798	0.805	**0.798**	0.731	0.675	0.591
CLWE fixed + NNmap	0.798	0.803	0.784	0.748	0.665	0.556
CLWE opt (LP)	0.797	0.804	0.779	0.725	0.655	0.605
CLWE opt (LLM)						
$k = 1$	0.813	0.811	0.764	0.731	0.680	0.569
k tuned to task	**0.815**	**0.812**	0.785	0.759	0.684	0.616
k tuned to task/language	**0.815**	0.810	0.777	**0.766**	**0.719**	**0.617**
Monolingual	0.879	0.857	0.838	-	-	-

Table 4: Classification accuracy of sentiment analysis task in cross-lingual settings. The underlined values indicate that, among the three trials, the worst model of **CLWE opt (LLM)** outperforms the best model of **CLWE fixed**.

CLWE fixed + NNmap adds two embedding-wise hidden layers to the original feedforward neural network in **CLWE fixed**. This is aimed at giving the network the capability of acquiring task-specific word embeddings by enhancing the representation of the network.

CLWE opt (LP) is **CLWE fixed** with embedding layer updated; we made this model cross-lingual by the linear projection (§ 3.2).

CLWE opt (LLM) is **CLWE fixed** with the embedding layer updated; we made this model cross-lingual by the locally linear mapping (§ 3.2). We report results with the three strategies to tune the hyperparameter k for cross-task projection.

Monolingual has the same network as **CLWE fixed** with the embedding layer updated; we trained the model with datasets in the same languages as testing. We present this result to show the upper bound of model accuracy.

The dimensions of all the layers of the above five models are 300, and they are all optimized by Adam optimizer (Kingma and Ba, 2014) for training. We conduct all experiments three times with different initialization of the model parameters and report the average accuracy, and hyperparameter tuning is conducted independently to each model.

Method	Topic Class.	Senti. Analysis	
		Amazon	Yelp
Monolingual fixed	0.921	0.828	0.799
Monolingual	**0.980**	**0.872**	**0.866**

Table 5: Classification accuracy of monolingual models in English.

5 Results

We evaluate the models in cross-lingual settings to confirm how well our method produces task-specific cross-lingual word embeddings (Table 3 and Table 4). Prior to reporting the results, we confirm the impact of task-specific word embeddings in neural networks through experiments in a monolingual setting in English (Table 5).

5.1 Impact of task-specific word embeddings

We examine the impact of optimizing the embedding layer of a neural network to the given task on model accuracy through experiments in

General	Topic class.	Senti. analysis (Amazon)
excellent		
excellently	excellently	awesome
superb	exceptional	perfect
good	tabcorp	pleased
impressive	novorossiisk	timeless
commendable	southcorp	mesmerizing
terrible		
horrible	frightening	horrible
dreadful	devastating	useless
awful	shocking	wasted
horrendous	mishaps	miserably
horrific	ugliness	refund
economic		
economy	imf	addition
macroeconomic	trade	nightstand
economies	economy	finances
microeconomic	wto	everyday
socio	economist	arguably

(a) English

General	Topic class.	Senti. analysis (Amazon)
excellent$_{excellent}$		
excellente$_{excellent}$	excellents$_{excellent}$	excellente
excellents	excellente	excellents
bon$_{good}$	excellentes$_{excellent}$	excellentes
excellentes	appréciable$_{appreciable}$	extraordinary
exceller$_{to\ excel}$	bons	parfaite
terrible$_{terrible}$		
terribles$_{terrible}$	terribles	terribles
horrible$_{horrible}$	horrible	horrible
terriblement$_{terribly}$	meurtrie$_{wounded}$	débile$_{stupid}$
épouvantable	gwynplaine	horribles$_{horrible}$
effroyable$_{terrifying}$	épouvantes$_{terrified}$	stupide$_{stupid}$
économie$_{economy}$		
economie$_{economy}$	économique$_{economic}$	economie
économique	économiques$_{economic}$	economiques
macroéconomie$_{macroeconomy}$	conjoncture$_{conjuncture}$	economic$_{economic}$
géoéconomie$_{geoeconomy}$	fmi$_{IMF}$	economique$_{economic}$
microéconomie$_{microeconomy}$	economique$_{economic}$	economies$_{economies}$

(b) French (English translations are given as subscripts)

Table 6: Nearest neighbors of some **words** in the semantic space of general and task-specific word embeddings.

English by comparing **Monolingual** to **Monolingual fixed** which is the same network as **Monolingual** with the embedding layer fixed to general words embeddings. We show the results of topic classification and sentiment analysis tasks in Table 5. In both tasks, **Monolingual** outperformed **Monolingual fixed** with a wide margin, which indicates that task-specific word embeddings are indeed crucial to obtain better model performance. This result motivates us to learn task-specific cross-lingual word embeddings to exploit the fully task-specific neural network.

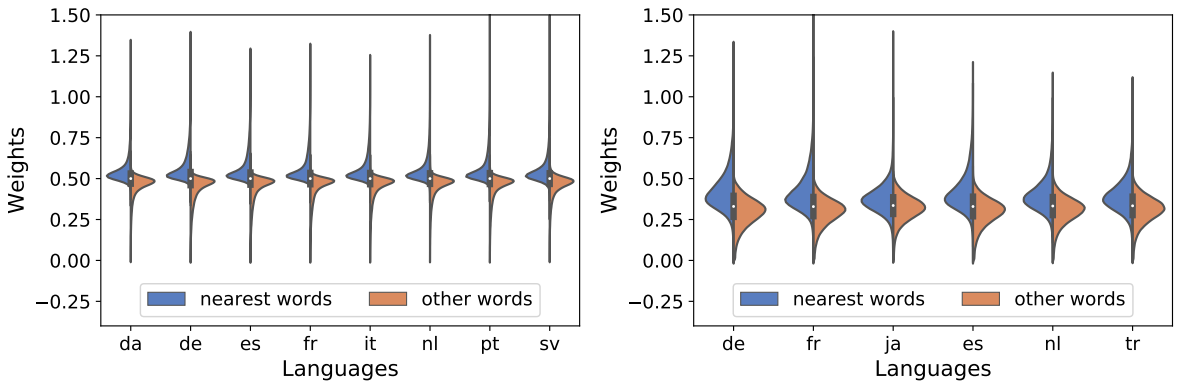
5.2 Performance of multilingual models

Table 3 and Table 4 report the classification accuracy of the models on topic classification and sentiment analysis, respectively. All models are trained in English and evaluated in the target languages. **CLWE opt** with hyperparameter k tuned on the source language successfully outperformed the two baselines, **CLWE fixed** and **CLWE fixed + NNmap**, in all task-language pairs except for English-German in the topic classification task and English-Japanese in the sentiment analysis task. This result indicates the importance of task-specific word representation in the multilingual model and that our projection successfully induced task-specific cross-lingual word embeddings. Although we gained some improvements by tuning k to the target language using the minimal development set in some configurations, the gains are smaller than the gains over the two baselines. This implies that k is more sensitive to the target task rather than the target language, which we discuss further in § 5.3.

In some languages, **CLWE fixed + NNmap** has even lower classification accuracy than **CLWE fixed**. We hypothesize that by having more layers, the model becomes more sensitive to the small difference in word representation, which means that the noise in pre-trained cross-lingual word embeddings affects on the model accuracy.

Comparing **CLWE opt (LLM)** to **CLWE opt (LP)**, we found that our locally linear mapping outperforms the linear projection method for a cross-task embedding projection. For some configurations, the performance of **CLWE opt (LP)** degrades significantly. These results indicate that the topology of the general and task-specific embedding spaces are so apart from each other that simple projection methods such as the linear projection are inappropriate. We will further discuss the difference in the topologies of the general and task-specific embedding spaces in § 5.3 by looking into nearest neighbors of some target words in the semantic space of general and task-specific cross-lingual word embeddings (Table 6).

In all configurations where sufficient dataset is available in the target languages, **monolingual** outperformed cross-lingual models with a wide margin. This indicates that there is still space for improvements in cross-lingual models.



(a) Topic classification ($k = 2$)

(b) Sentiment analysis ($k = 3$)

Figure 2: Distribution of the reconstruction weights $\hat{\alpha}$ for the nearest words of the target words and the other nearest neighbors.

5.3 Analysis

We conduct further investigation to gain a profound understanding of our method and the resulting task-specific cross-lingual word embeddings. We first analyze the task-specific cross-lingual word embeddings through nearest neighbors of some words. We next investigate the distribution of the reconstruction weights to see the impact of k nearest neighbors other than the nearest one. We then evaluate the sensitivity of the model accuracy to the value of k.

Properties of task-specific embeddings Here, we examine the properties of task-specific word embeddings obtained using our cross-task projection. For this purpose, we present nearest neighbors of frequent words in the tasks in various embeddings in English and French.

Table 6a shows nearest neighbors of "excellent," "terrible," and "economic" in the general word embeddings, and the embedding layer of the models optimized for the training data in English. In the general embeddings, the words are close to words that have similar semantic or syntactic while the task-specific word embeddings show different properties specific to the target tasks.

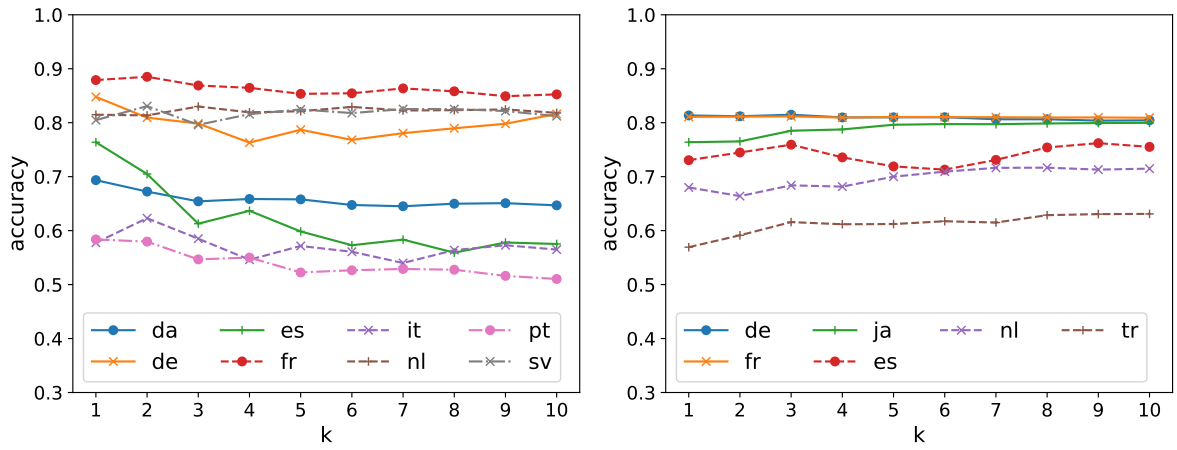
In the embedding layer optimized for topic classification, we found "economic" to be close to "imf (International Monetary Fund)" or "wto (World Trade Organization)." Even though they are semantically distinct, they all strongly indicate the Economy label. In contrast, the nearest neighbors of "excellent" and "terrible" are noisy since they do not contribute to the topic classification task.

The embedding layers optimized for sentiment analysis exhibit different properties. While the nearest neighbors of "excellent" and "terrible" are not semantically close, they all indicate positive and negative polarities in the respective domains. However, the nearest neighbors of "economic" are noisy as they do not contribute to the task.

Table 6b shows nearest neighbors of "excellent (*excellent*)," "terrible (*terrible*)," and "économie (*economy*)" in French; the general word embeddings (**General**) and the task-specific word embeddings obtained using our cross-task projection (**LLM**). **General** embeddings exhibit similar properties as English ones.

LLM embeddings of topic classification task have "fmi (*IMF; International Monetary Fund*)" and "conjoncture (*conjuncture*)" as nearest neighbors of "économie." This indicates that our cross-task projection successfully obtains word embeddings optimized for the task since they are strong signals of the Economy label. For sentiment analysis, the word embeddings obtained by our cross-task projection of Amazon dataset captures "extraordinary" and "parfaite," which strongly indicate positive polarity, as the nearest neighbors of "excellent" In contrast, the words strongly associated with negative polarity, "débile" and "stupide," are the nearest neighbors of "terrible" in the embedding space. These properties suggest that our cross-task projection successfully obtains task-specific cross-lingual word embeddings.

Distribution of the reconstruction weights To see how much the nearest neighbors for the target words contribute to the projection, we investigate the distribution of $\hat{\alpha}$ induced by Eq. 3. Figure 2



(a) Topic classification

(b) Sentiment analysis

Figure 3: Classification accuracy as a function of k in cross-task embedding projection.

shows the distribution of the absolute value of $\hat{\alpha}$ for the nearest neighbors of the target word and the other nearest neighbors. For this experiment, we used k tuned on the source language.

Even though the nearest words tend to have a slightly higher value of $\hat{\alpha}$ compared to the other nearest neighbor words, the difference is not so significant for most of the configuration. This observation indicates that all of the k-nearest neighbors contribute to the projection.

Sensitivity to hyperparameter k We proposed three strategies to tune the hyperparameter k of our locally linear mapping for cross-task embedding projection of cross-lingual word embeddings: tuning on the development data in the source language as described in § 3.2, preparing small development data (100 samples) in the target languages, or fixing $k = 1$. Revisiting results in Table 3 and Table 4, for the topic classification task, the classification accuracy of the models are consistent among all of the tuning methods (Table 3), while for the sentiment analysis task, fixing $k = 1$ yields lower classification acuracy (Table 4). Here, we conduct further analysis to gain a profound understanding of the effect of the value of k.

Figure 3 depicts the classification accuracy of the models on the test set while varying k in the topic classification task and sentiment analysis task. Across languages, a smaller value of k yields better performance for the topic classification task, while a larger value of k yields better performance for the sentiment analysis task. These results indicate that the best value of k is language-independent and thus, the tuning k for the development set of source language suffices to achieve good results.

6 Conclusions

We proposed a method to obtain a fully task-specific multilingual model without relying on any cross-lingual resources or annotated corpora in the target language by a cross-task embedding projection. Because a naive linear projection puts too strong assumption on the topologies of two embedding spaces, we present an effective method for the cross-task embedding projection named locally linear mapping. The locally linear mapping assumes and preserves the local topology across the semantic spaces before and after the projection. Experimental results demonstrated that the locally linear mapping successfully obtains task-specific word embeddings of the target language, and the resulting fully task-specific multilingual model exhibited better model accuracy than the existing multilingual model that fixes its embedding layer to general word embeddings.

We plan to evaluate our method on various NLP tasks, languages, and neural network models, and investigate the results to devise an adaptive method to tune k for individual words.

Acknowledgements

We deeply thank Satoshi Tohda for proofreading the draft of our paper. We also thank Dr. Junpei Komiyama for checking the mathematics. This research was supported by NII CRIS Contract Research 2019.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 789–798.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146.

Ethem F. Can, Aysu Ezen-Can, and Fazli Can. 2018. Multilingual sentiment analysis: An RNN-based framework for limited data. In *ACM SIGIR 2018 Workshop on Learning from Limited or Noisy Data (LND4IR)*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics (TACL)*, 6:557–570.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. 2017. Multilingual training of crosslingual word embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 894–904.

Yanlin Feng and Xiaojun Wan. 2019. Learning bilingual sentiment-specific word embeddings without cross-lingual supervision. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 420–429.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1386–1390.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC)*, pages 3483–3487.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for POS tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2832–2838.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the third International Conference on Learning Representations (ICLR)*.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fei Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5(Apr):361–397.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Ge Xu, and Houfeng Wang. 2012. Cross-lingual mixture model for sentiment classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 572–581.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *Computing Research Repository, arXiv:1309.4168. Version 1*.

Nikolaos Pappas and Andrei Popescu-Belis. 2017. Multilingual hierarchical attention networks for document classification. In *Proceedings of the eighth International Joint Conference on Natural Language Processing (EACL)*, pages 1015–1025.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphee De Clercq, Veronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Núria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval)*, pages 19–30.

Sam T. Roweis and Lawrence K. Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2019. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research (JAIR)*, 65:569–631.

Shyam Upadhyay, Manaal Faruqui, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2018. (Almost) zero-shot cross-lingual spoken language understanding. In *Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6034–6038.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the fourth International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 235–243.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pages 1006–1011.

Kui Xu and Xiaojun Wan. 2017. Towards a universal sentiment classifier in multiple languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 511–520.

Zhilin Yang, Ruslan Salakhutdinov, and William W. Cohen. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. In *Processing of the fifth International Conference on Learning Representations (ICLR)*.

A Derivation of the locally linear mapping

Recall that X^{gen} and Y^{gen} represent general cross-lingual word embeddings of the source and target languages, respectively. Also, for each word i in the target language, we denote the set of its k nearest neighbors in the target language in the semantic space of the general cross-lingual word embeddings as $\mathcal{N}_i^{\text{gen}}$.

We reconstruct Y_i^{gen} as a linear combination,

$$\sum_{j \in \mathcal{N}_i^{\text{gen}}} \alpha_{ij} X_j^{\text{gen}}$$

where $\boldsymbol{\alpha}_i$ is the weight vector which we optimize. The reconstruction error is given as

$$\begin{aligned}
\epsilon &= \left\| Y_i^{\text{gen}} - \sum_{j \in \mathcal{N}_i^{\text{gen}}} \alpha_{ij} X_j^{\text{gen}} \right\|^2 \\
&= \left\| \sum_{j \in \mathcal{N}_i^{\text{gen}}} \alpha_{ij} \left(Y_i^{\text{gen}} - X_j^{\text{gen}} \right) \right\|^2 \\
&= \sum_{j \in \mathcal{N}_i^{\text{gen}}} \sum_{i \in \mathcal{N}_i^{\text{gen}}} \alpha_{ij} \alpha_{il} C_{ijl}
\end{aligned}$$

where $C_i \in \mathcal{R}^{k \times k}$ is the covariance matrix,

$$C_{ijl} = \left(Y_i^{\text{gen}} - X_j^{\text{gen}} \right) \left(Y_i^{\text{gen}} - X_l^{\text{gen}} \right).$$

We minimize this reconstruction error ϵ under the constraint of $\sum_{j \in \mathcal{N}_i^{\text{gen}}} \alpha_{ij} = 1$. Applying the method of Lagrange multiplier, we have

$$L = \sum_{j \in \mathcal{N}_i^{\text{gen}}} \sum_{i \in \mathcal{N}_i^{\text{gen}}} \alpha_{ij} \alpha_{il} C_{ijl} - \lambda \left(\sum_{j \in \mathcal{N}_i^{\text{gen}}} \alpha_{ij} - 1 \right).$$

We then solve $\frac{\partial L}{\partial \alpha_{ij}} = \frac{\partial L}{\partial \lambda} = 0$ to obtain

$$\hat{\alpha}_{ij} = \frac{\sum_l (C_i^{-1})_{jl}}{\sum_j \sum_l (C_i^{-1})_{jl}}.$$

The resulting value of $\hat{\boldsymbol{\alpha}}_i$ is then used to compute the task-specific word embedding of i as

$$Y_i^{\text{spec}} = \sum_{j \in \mathcal{N}_i^{\text{gen}}} \hat{\alpha}_{ij} X_j^{\text{spec}}$$

where X^{spec} is the tast-specific word embeddings of the source language.

Investigating Cross-lingual Alignment Methods for Contextualized Embeddings with Token-Level Evaluation

Qianchu Liu, Diana McCarthy, Ivan Vulić, Anna Korhonen
Language Technology Lab, University of Cambridge
English Faculty Building, 9 West Road, Cambridge CB3 9DA, United Kingdom
{ql261,iv250,alk23}@cam.ac.uk, diana@dianamccarthy.co.uk

Abstract

In this paper, we present a thorough investigation on methods that align pre-trained contextualized embeddings into shared cross-lingual context-aware embedding space, providing strong reference benchmarks for future context-aware crosslingual models. We propose a novel and challenging task, Bilingual Token-level Sense Retrieval (BTSR). It specifically evaluates the accurate alignment of words with the same meaning in cross-lingual non-parallel contexts, currently not evaluated by existing tasks such as Bilingual Contextual Word Similarity and Sentence Retrieval. We show how the proposed BTSR task highlights the merits of different alignment methods. In particular, we find that using context average type-level alignment is effective in transferring monolingual contextualized embeddings cross-lingually especially in non-parallel contexts, and at the same time improves the monolingual space. Furthermore, aligning independently trained models yields better performance than aligning multilingual embeddings with shared vocabulary.

1 Introduction

Contextualized embeddings have been shown to achieve superior performance compared to static word embeddings in English (Peters et al., 2018; Devlin et al., 2019). Despite recent efforts to better understand their multilingual variants (Pires et al., 2019), leveraging these available pretrained contextualized embeddings to learn cross-lingual contextualized embeddings is still an under-explored area: past cross-lingual embedding alignment methods have mainly focused on static embeddings (Ruder et al., 2019). In this paper, we introduce a first study that investigates and compares different ways of aligning the pretrained contextualized embeddings. In particular, we make the comparisons focused on the following properties: (1) aligning contextualized embeddings at the level of word tokens versus word types; (2) different training signals: static dictionaries, word alignment, or sentence alignment from parallel data; and (3) aligning different model variants: aligning from independently trained models versus aligning embeddings from a multilingual model with shared vocabulary.

We evaluate the methods on a variety of context-aware tasks. Besides two previously established evaluation tasks (1) Bilingual Contextual Word Similarity (Chi and Chen, 2018) and (2) Sentence Retrieval (Conneau et al., 2017), we introduce a new task: Bilingual Token-level Sense Retrieval (BTSR). It is more challenging than the alternatives as it requires the accurate cross-lingual retrieval of contextualized words on the token level which are disambiguated both in the source and the target language using non-parallel contexts. We provide BTSR task data and run evaluations on two language pairs: English–Chinese (EN–ZH) and English–Spanish (EN–ES). The data and guidelines can be found at: https://github.com/qianchu/BTSR

Our main findings are as follows. (1) Using the average of the contextualized word representations as type-level anchors is effective and robust for aligning pre-trained contextualized embeddings cross-lingually, and can also improve the monolingual contextualized space as it brings the largest gains in English context-aware evaluation compared to results from aligning on other levels. (2) Using a dictionary with a few thousand entries is able to yield performance comparable to leveraging training signals from parallel corpora. (3) Aligning independently trained models performs better than aligning embeddings from a multilingual model trained with shared vocabulary.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 33–43
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

2 Related Work

Cross-lingual Word Embeddings. We conduct our experiments using a popular projection-based approach that learns an orthogonal mapping between pretrained embeddings (Xing et al., 2015; Artetxe et al., 2016). The orthogonality of the mapping is crucial as it preserves monolingual invariance and is empirically proven to be more robust (Smith et al., 2017; Xing et al., 2015). This projection-based method can be applied post-hoc on pretrained monolingual embeddings with an exact analytical solution. Moreover, its performance is often competitive to that of jointly trained cross-lingual models using additional bilingual signals in the form of parallel or comparable corpora (Ruder et al., 2019; Glavaš et al., 2019).

However, projection-based cross-lingual embeddings are still predominantly concerned with static word embeddings (Glavaš et al., 2019; Vulić et al., 2019; Mohiuddin and Joty, 2019). Learning cross-lingual contextualized embeddings is still a large unexplored area with only two concurrent papers at the moment. First, Aldarmaki and Diab (2019) adopt the same projection-based approach as our paper to align contextualized embeddings on the token-level using parallel data. They find that context-aware mapping using parallel data outperforms context-independent mappings from static dictionaries on a parallel Sentence Retrieval task. Second, Schuster et al. (2019) introduce anchor embeddings as the average of contextualized embeddings of a word to perform alignment for contextualized models, and show its effectiveness in cross-lingual dependency parsing. These two studies are not directly comparable, whereas our paper provides a comprehensive and systematic comparison of various methods for learning cross-lingual contextualized embeddings and introduces a new and more challenging evaluation task.

Evaluation of (Contextualized) Cross-lingual Embeddings. The traditional task to evaluate cross-lingual embeddings is Bilingual Dictionary Induction (BDI) (Vulić and Moens, 2013; Mikolov et al., 2013a; Gouws et al., 2015): given a source query word, the task is to retrieve the translation word in the target language. The test words in BDI are out-of-context and polysemy cannot be addressed properly. The same issue is found in another relevant lexical task, Cross-lingual Semantic Similarity. (Camacho-Collados et al., 2017).

The only context-aware dataset for evaluating cross-lingual embeddings on the word level is Bilingual Contextual Word Similarity (BCWS) (Chi and Chen, 2018). It challenges a system to predict similarity scores between cross-lingual word pairs with sentential context provided in both languages. However, BCWS does not explicitly test for the retrieval of meaning-equivalent cross-lingual contextualized embeddings, which is explicitly tested in our test. Also, BCWS is only available for one language pair: English-Chinese.

Another task used for evaluating contextualized embeddings is Sentence Retrieval (Aldarmaki and Diab, 2019): given a query source sentence, the task is to retrieve the corresponding parallel sentence in the target language. Sentences can be represented as averages of contextualized embeddings of their constituent words. As the task does not explicitly evaluate at the word level, even if a system cannot accurately capture polysemy, it can rely on other words in the sentence to retrieve the correct parallel sentence. Therefore, Sentence Retrieval may lead to superficially high scores.

Cross-lingual Word Sense Disambiguation. Our new task is also related to Cross-lingual Word Sense Disambiguation (Lefever and Hoste, 2009): given a source language word in context, a system needs to provide the correct sense labels as clustered translation words in a number of target languages. Another related task is Cross-lingual Lexical Substitution (Sinha et al., 2009): the model must provide plausible target language translations for the source language lexical item in the source language context. In contrast, our BTSR task: (1) directly evaluates token-level word representations without the need to predict sense labels from a sense inventory and (2) it contextualizes both the source query and the target candidates ensuring full sense disambiguation. The core differences between the three tasks are illustrated in the following examples below:

(1) Cross-lingual Word Sense Disambigution:
source query: the national [coach] of the Irish teams ...
answer: allenatore (Italian); Fußbaltrainer; Nationaltrainer; Trainer (German); entrenador(Spanish) ...

(2) Cross-lingual Lexical Substitution :
source query: She looked as [severely] as she could muster at Draco.
answer: rigurosamente, seriamente

(3) BTSR:
source query: The reflections included in this document are linked to discussions with many colleagues and friends, in the present [tense].

answer: Scott Peterson metió la pata elfondo y usó el [tiempo] pasado mientras afirmaba que su esposa asesinada estaba viva , lanzando una búsqueda (...)

3 Methods

3.1 Monolingual Contextualized Embeddings

Compared to static word embeddings (Mikolov et al., 2013b; Bojanowski et al., 2017), more recent contextualized embeddings provide dynamic representations for a word in context as hidden layers in a deep neural network. They are typically obtained by unsupervised pretraining based on language modeling objectives (Devlin et al., 2019; Yang et al., 2019). The underlying contextualized method in our study is the pretrained $BERT_{base}$ cased model[1] (Devlin et al., 2019). BERT is trained using a transformer architecture (Vaswani et al., 2017) with masked language modelling (MLM) and next sentence prediction (NSP) tasks. MLM predicts the vocabulary id of a randomly masked word in a sentence based on the word's context. NSP trains text-pair representations to predict whether the text-pair contains consecutive sentences from a monolingual corpus.[2]

We work with two BERT variants. First, we explore aligning independently trained BERT models, that is, models with separate model parameters for each language. For English and Chinese, we align independently trained Chinese and English monolingual models. For Spanish and English, since there is no pretrained BERT Spanish model, we take the Spanish embeddings from the BERT multilingual model and align it with the monolingual English model. We take this alignment as an approximation to aligning two independently trained models. We have also experimented with directly aligning embeddings obtained from the BERT multilingual model, which is a joint model trained with the same model parameters with shared subword vocabulary (Devlin et al., 2019). This means that identical words in two different languages will obtain the same embeddings.

[1]To produce the contextualized representation for a word in context, we average the 12 hidden layers of the word's subword representations in BERT and then average the subword representations as input for the cross-lingual alignment. We leave other ways to extract the representations for future work.

[2]We have also experimented with ELMo in lieu of BERT (Peters et al., 2018; Che et al., 2018). However, as we reach similar conclusions in terms of relative performance, while BERT-based cross-lingual embeddings outperform their ELMo-based counterparts in absolute terms, we do not report ELMo's results for brevity. It should be noted that these pretrained models used different training data.

3.2 Orthogonal Mapping and MIM

Given a dictionary with item pairs from source and target languages (s_i, t_i), and matrices S and T that contain the vector representations corresponding to the item pairs in the columns, we follow the standard practice (Glavaš et al., 2019) to find an orthogonal alignment matrix W that minimizes the distance between the transformed matrix WS and T. For improved performance, following Artetxe et al. (2016), we normalize and mean center the embeddings in S and T. The mapping is as follows:

$$W = \arg\min_{W} \|WS - T\|^2 \quad s.t. \quad W^T W = I. \qquad (1)$$

The closed-form solution can be found by solving the orthogonal Procrustes problem (Schönemann, 1966) as follows:

$$TS^T = U\Sigma V^T; W = UV^T \qquad (2)$$

We also optionally apply a post-processing *Meeting-in-the-Middle* (MIM) technique, recently proposed by Doval et al. (2018). It first calculates the average of each dictionary item representation in a pair after the orthogonal mapping: we denote the matrix U as the matrix where each column is such an average vector. Then, it finds a linear mapping M from both the source language (denoted as M_s) and the target language (M_t) after the previous step of orthogonal mapping to minimize the distance to U via a closed-form solution. Equation (3) formulates how to find M_s, and we do the same from target to source.

$$M_s = \arg\min_{M_s} \|M_s WS - U\|^2 \qquad (3)$$

We apply the orthogonal mapping and MIM both on static embeddings (for baselines) and contextualized embeddings. For mapping the contextualized embeddings, we either extract type-level embeddings from the contextualized models to serve as anchors for the alignment using static dictionaries, or we use parallel sentences as dictionary items to directly align contextualized word representations on the token level. We discuss this in what follows.

3.3 Alignment Levels

We explore aligning contextualized models on two levels: *type-level* and *token-level*. Type-level word representation refers to static word representation that assigns one fixed embedding to a word. All the traditional word embedding models (e.g., skip-gram, CBOW, fastText) provide such embeddings, and cross-lingual alignment is typically applied on

these type-level embeddings (Ruder et al., 2019). Token-level word representation refers to dynamic representations for words *in context*, i.e., contextualized word representations.

Contextualized models such as BERT provide token-level embeddings by default: a natural way to align these embeddings is token-level alignment. This has been proposed concurrently to our work by Aldarmaki and Diab (2019). This method requires token-level training data , e.g., from a word-aligned parallel corpus.

As an alternative, we obtain static type-level representations in the same space as our contextualized embeddings and use these type-level representations as anchors to learn the crosslingual mapping. The type-level anchors can be seen as taking a representative sample of the infinite space of the contextualized embeddings. The mapping learned via the anchors will hopefully be generalizable to align the dynamic token-level contextualized embeddings as well. The advantage of this approach is that we can align the contextualized embeddings with a standard dictionary now that we have one representation per word.

We experiment with two different kinds of anchor type-level embeddings: iso_type and avg_type. The iso_type refers to type-level embeddings that are produced by simply inputting the word in isolation to the contextualized model. Avg_type embeddings are obtained by taking the average of the contextualized representations of a word.[3] The context-average avg_type embeddings has been proposed recently by Schuster et al. (2019). In this work, we provide a systematic comparison of embeddings aligned on the token level, and on the two kinds of type-level alignments.

3.4 Alignment Training Signal

We explore a number of different supervision signals for learning the alignment between monolingual embeddings. First, we evaluate traditional methods that exploit word-level training signals (Ruder et al., 2019). We use (1) a static manually created (i.e., external) dictionary to obtain the alignment, and (2) we rely on word alignments from a parallel corpus as the source of the training signal. For word alignments, we either treat them as a large dictionary to perform type-level alignment or we additionally leverage the context in the aligned sentences to extract a dynamic contextualized dictionary to perform token-level alignment.

We also exploit the training signal coming from the aligned parallel sentences alone without word alignments. We first create sentence representations by averaging type-level or token-level embeddings, and then align the parallel sentence representations from source to target language.

The configurations for learning cross-lingual contextualized word embeddings explored in this work are summarized in Table 1, and we rely on the configuration labels from the table throughout the paper. Type-level configurations which ignore context are treated as baselines.

4 Bilingual Token-level Sense Retrieval Task (BTSR)

Task Description. In §2, we already discussed the main properties of the two other tasks that can be used to evaluate cross-lingual context-aware embeddings: BCWS and parallel Sentence Retrieval. In short, BCWS only measures similarity between cross-lingual word pairs in context, and it does not evaluate the translation capacity of different methods. The Sentence Retrieval task does not evaluate on the word level and can be solved by relying on the context alone.

To bridge this gap in evaluation, we introduce a new task: Bilingual Token-level Sense Retrieval (BTSR). It tests for the retrieval of meaning-equivalent cross-lingual contextualized word embeddings relying on non-parallel context information. Our task can be seen as a contextualized variant of the BDI task. Its comparison to the traditional BDI task is provided in Table 2.

In what follows, we define the BTSR task formally and provide details on how the task data is created. To build a representative sample of contextualized words in the source and target languages, we collect translation pairs and contextualize the word pairs into token-level representations. Then we manually check a sample of the contextualized word pairs to ensure correspondence of sense on the token-level. To understand the effect of the size of the search space, we experiment with 20k and 200k candidates respectively.

Formal Definition. In BTSR, we define S : $s^1_{tk,1}, s^1_{tk,2}, s^2_{tk,1}, \ldots, s^n_{tk,m}$ as a set of queries from the source language. A query $s^i_{tk,j}$ is a token-level contextualized representation of the ith source

[3] In practice, we take 1000 random samples for a word from the training data of the parallel corpora used in our experiments.

Component	Options	Label
Alignment Signal	Word alignment from parallel data	wa
	Sentence alignment from parallel data	sa
	MUSE training dictionary	dict
Alignment Level	Token-level alignment	token
	Type-level alignment from context average	avg_type
	Type-level alignment from inputting the word in isolation	iso_type
	Type-level alignment in static embeddings (eg. Fasttext)	type
Models	monolingual English BERT model	mono_en
	monolingual Chinese BERT model	mono_zh
	BERT multilingual English model	multi_en
	BERT multilingual Spanish model	multi_es
	Fasttext baseline	fasttext
Alignment techniques	the original orthogonal linear transformation	orig
	post-processing linear transformation after the orthogonal transformation	mim
Evaluation level	Evaluated on token-level representations	[token]
	Evaluated on type-level representations	[type]

Table 1: Different components used for the model configurations in our evaluation.

BDI		BTSR	
uniform	制服	..[uniforms] were black...	他的[制服].. (His [uniform]..)
subdue	制服	..mosquito was [subdued]..	..[制服]刺客.. (...[subdue] the assassin...)
uniform	一致	the [uniform] convergence of the regular solution	...[一致] 漸近穩定...定理 (the theorem of [uniform] asymptotic stability...)

Table 2: BTSR: examples and a comparison with traditional (non-contextualized) BDI.

word that corresponds to the word's jth sense. Similarly, we define $T : t^1_{tk,1}, t^1_{tk,2}, \ldots, t^p_{tk,q}$ as a set of candidates in the target language where each candidate is a contextualized token-level word that represents a specific sense of a word in the target language. For each query s_{tk}, the task is to find a target contextualized token-level word t_{tk} that has the same word sense as in the query. $Sim(s_{tk}, t_{tk})$ is a function that computes the similarity of s_{tk} and t_{tk}. In our experiments, we use cosine similarity. Using $Sim(s_{tk}, t_{tk})$, for each query, we retrieve $t_{tk,i^1}, \ldots, t_{tk,i^K}$: the top K most similar token-level contextualized words from the target set T in the cross-lingual space as the nearest neighbours. We report *Precision@K*, i.e. precision of finding the gold t_{tk} in the top K retrieved candidates.

Collecting Translation Pairs. We select a representative set of query words from WordNet (Miller, 1998) (one unique word per WordNet synset). For each source word, we retrieve its WordNet senses and the corresponding translations in the target language from Multilingual WordNet (Bond and Foster, 2013). As WordNet senses are too fine-grained, we collapse senses into clusters if they contain the same translation for the source word. For example, "uniform" has five WordNet senses which are translated into four distinct Chinese words: 制服(the clothes worn by a particular group), 一致(the translation of two senses: consistent and undifferentiated)[4], 不變(unchanged) and 相同(the same) . We take these four Chinese words to form four translation pairs with "uniform".

Word Pair Contextualization. For each word in a word pair, we "contextualize" the word by selecting a sentence in which the word appears, and ensure that the resulting contextualized word can be translated into the other word. Therefore, if a polysemous word occurs in multiple word pairs with distinct translations, it will be accompanied with different contexts that correspond to each translation. We achieve this by selecting a pair of parallel sentences in which the source word and the target word from the word pair are aligned after we run word alignment. The context in the source language in this parallel sentence pair is used to "contextualize" the source word. When we select context for the target word, we choose a different parallel sentence in which the two words in the pair are aligned. Therefore, the final contexts for the source and target word in the word pair are indeed non-parallel.

The use of non-parallel contexts here is crucial because when we perform the token retrieval task, parallel contexts can be superficially retrieved by simply matching the contexts rather than repre-

[4]Notice the senses are different thus contexts are needed to find the pair corresponding to the same meaning.

senting the words in context appropriately. We empirically verified that a simplistic context average baseline outperforms contextualized word embeddings in a variant of our task which relies on parallel contexts.

We set aside 1M parallel sentences from the UMCorpus (Tian et al., 2014) (EN–ZH) and the WMT13 news dataset (Bojar et al., 2013) (EN–ES) for extracting the sentence contexts. We end up with 14,604 distinct word pairs with contexts extracted for EN–ZH, and 9,623 pairs for EN–ES.

Creation of Test Data. As the contexts are non-parallel in a word pair, we need to check if the contextualized words in a word pair genuinely represent the same meaning. We manually checked a sample of the word pairs extracted in the previous step to produce the final test set for BTSR. To produce the sample, we selected the translation pairs that satisfy any of the following constraints: 1) target or source word belongs to the top 250 frequent words in each language, 2) target or source word belongs to the top 250 most ambiguous words in each language. We take the number of sense clusters as introduced above as a measure of ambiguity for each word.

The first author then provided an initial manual annotation of the samples for both EN–ES and EN–ZH on whether the contextualized words in a pair correspond to the same meaning. The samples from the two language pairs were subsequently annotated by one native Chinese speaker and one native Spanish speaker respectively. The final agreement rate calculated as pairwise inter-annotator agreement on a binary choice[5] for EN–ZH is 94.5%, and 94.7% for EN–ES. Finally, we take the subsets where all annotators agree as the test sets for EN–ZH (1,181 pairs) and EN–ZH (994 pairs).

Target Candidates. We treat the token-level representations of the target words from all words pairs in the contextualization process described above as our candidate space. To make the target candidate space more representative of the language, we supplement the space with words outside of the WordNet inventory from monolingual Wikipedia dumps in the target language. For each of these words, we randomly select a sentence in which it occurs to contextualize the word into a token-level target candidate. We experiment with 20k target candidates and 200k target candidates.

[5]For each language pair, it is calculated as the percentage of token pairs marked correct by both annotators (the first author and one native speaker of the language) divided by the number of all the token pairs.

5 Experiments

Training Setup. To test the effects of corpora size on the induction of the cross-lingual alignment, we vary the size of the parallel corpus from 100 up to 200k parallel sentences in the UMCorpus and the WMT13 corpus. Word alignment was produced by IBM Model 2 using Fastalign (Dyer et al., 2013). We also induce cross-lingual alignments relying on static dictionaries provided by MUSE (Conneau et al., 2017). BERT variants (see §3.1) are taken from Devlin et al. (2019). For comparison with BERT, we also run fasttext (Bojanowski et al., 2017) to produce baseline static embeddings using the same training Wikipedia corpora for English, Chinese and Spanish.

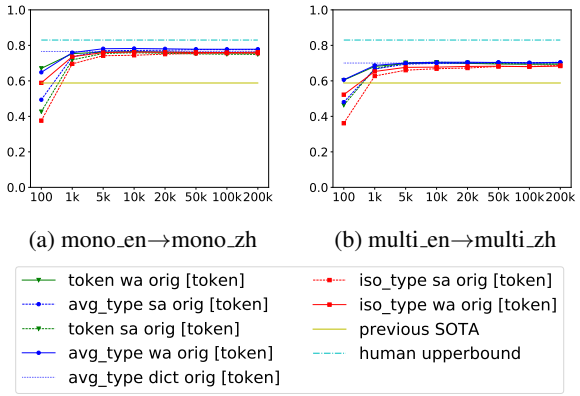
5.1 Bilingual Contextual Word Similarity

We first evaluate the models on two previous evaluation tasks: BCWS and Sentence Retrieval. For both tasks, we compute cosine similarity to measure the distance between representations. For BCWS, we evaluate embedding distance against human annotations via Spearman correlation. Results on the BCWS task for EN–ZH are shown in Figure 1. The main finding is that all cross-lingual contextualized embeddings in our comparison surpass the previous state-of-the-art (SOTA) based on a cross-lingual multi-sense model (Chi and Chen, 2018) as soon as they are fed 5K or more parallel sentences. Note that the previous SOTA model was trained on the full EN–ZH parallel corpus of around 2M sentences. Although BERT was pretrained on a corpus comprising 3.3B words , it is reasonable to assume that it is easier to procure abundant monolingual data than parallel data. Therefore, aligning pretrained monolingual embeddings using only a small amount of parallel data rather than training on a large parallel corpus is a more favorable choice.

Alignment based on independent monolingual models (mono_en→mono_zh) is particularly effective, achieving human-level performance. While different methods achieve comparable results, avg_type consistently takes the lead.

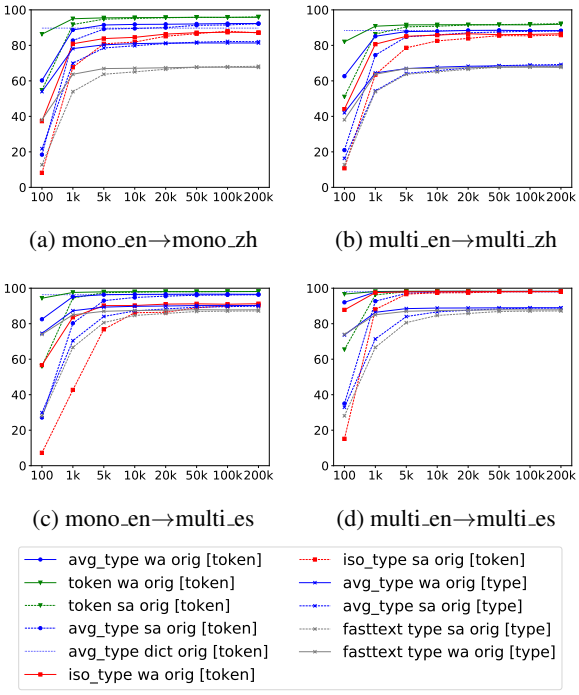
5.2 Sentence Retrieval

For the Sentence Retrieval task, we compute cosine similarity between the query sentence representation and sentence representations in the tar-



(a) mono_en→mono_zh (b) multi_en→multi_zh

Figure 1: BCWS (Spearman's ρ). The horizontal axis indicates the number of parallel sentences used for learning the alignment transformation. Please refer to Table 1 for understanding the method acronyms in the legend. For example, 'token wa orig [token]' refers to token-level orthogonal mapping trained with word alignment and it is evaluated on token-level data.



(a) mono_en→mono_zh (b) multi_en→multi_zh

(c) mono_en→multi_es (d) multi_en→multi_es

Figure 2: Results on the Sentence Retrieval task from the testset of UMcorpus and WMT13 corpus; the scores are *Precision@5* (%). The horizontal axis indicates the number of parallel sentences used for learning the alignment transformation. Please refer to Table 1 for understanding the method acronyms.

get language in the test set of UMcorpus (English-Chinese) and WMT13 corpus (English-Spanish). Precision results for finding the parallel sentence in the top 5 candidates are reported in Figure 2. We find that evaluating with contextualized embeddings on the token-level (all the *[token]* lines) performs consistently better than type embedding baselines. Among the different ways to transfer the contextualized embeddings, aligning directly on the token level with parallel data outperforms aligning via type-level anchoring. Concerning the alignment training signal, sentence alignment starts low but is able to yield comparable results with word alignment after 50K sentences. For the EN–ZH Sentence Retrieval, aligning independently trained BERT models outperforms aligning embeddings with shared vocabulary. For the EN–ES Sentence Retrieval task, aligning from both independent models and from shared embeddings achieves ceiling performance.

5.3 Bilingual Token-level Sense Retrieval

We report *Precision@5* scores for 20k target words in Figure 3. We also report the results from aligning using 200k parallel sentences on BTSR with 200k target words and applying the additional MIM technique in Table 3.

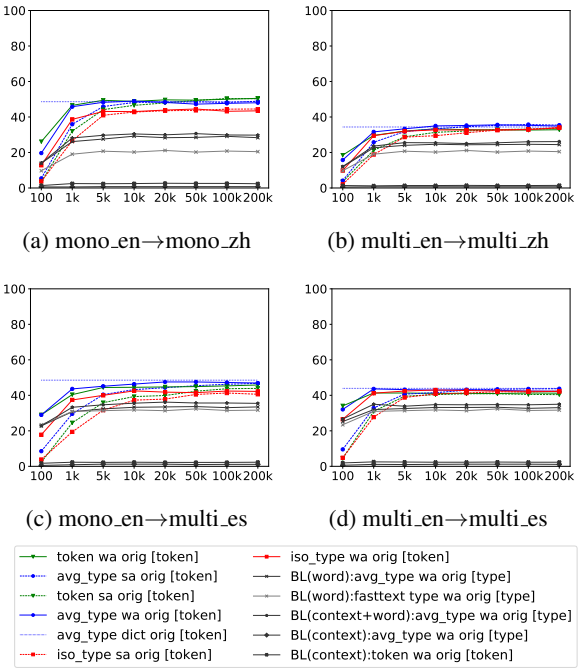
Baselines. We evaluate four baselines that help us better understand the models' performance in this task. For *BL(word)* methods, we discard the contexts and use only the query and target word's type representations. Therefore, polysemous words in the dataset will have only one static representation. We implement both a fasttext baseline and a context-average type embedding baseline for each contextualized model. We also provide baselines which use context but ignore the word in focus (*BL(context)*). These baselines take an average of the context embeddings both at the token level and at the type level of the contextualized models. Instead of finding the best translation word in context, these baselines retrieve the target sentence with the best translation of the source context.[6] Finally, we evaluate a simple baseline that combines both word and context as an average of the two representations. Context representation here is the average of the context embeddings. Both word and context embeddings here are calculated using the avg_type embeddings.

Discussion. The low performance of all the baselines suggest that the proposed task is more challenging than the alternatives: it can not be easily

[6] On our trivial parallel variant of the task, this context baseline gives the best performance.

	token		avg_type		iso_type	
	wa	sa	wa	sa	wa	sa
mono_en→mono_zh	30.84	28.87	**32.04**	31.7	25.43	26.46
+ mim	29.98	30.15	**34.79**	34.45	26.37	27.15
multi_en→multi_zh	17.14	16.97	19.9	**20.84**	16.8	18.17
+ mim	15.93	16.62	21.62	**21.79**	14.81	14.9
mono_en→multi_es	33.47	30.15	**34.37**	33.37	29.25	28.44
+mim	32.46	30.55	**35.38**	33.57	27.34	25.43
multi_en→multi_es	27.14	25.43	**29.35**	29.25	27.04	26.33
+mim	28.44	27.94	**31.86**	31.76	26.73	25.03

Table 3: BTSR results with 200k candidates; alignment learned from 200k parallel sentences. Please refer to Table 1 for the explanation of the acronyms.



(a) mono_en→mono_zh (b) multi_en→multi_zh

(c) mono_en→multi_es (d) multi_en→multi_es

Figure 3: EN–ZH and EN–ES BTSR results; *Precision@5* (%). The horizontal axis indicates the number of parallel sentences used for learning the alignment transformation. Please refer to Table 1 for understanding the method acronyms.

tackled by looking at word in isolation (i.e., at type-level representations) or the context alone, or a simple combination of context and the query word.

Regarding the alignment level, compared to the Sentence Retrieval task, the benefit of dynamic token-level alignment from parallel corpora now disappears. Aligning the contextualized embeddings via context-average anchor type embeddings, i.e. avg_type alignment, (which consistently outperform iso_type embeddings) is the best model in most cases, or yields comparable performance with token-level alignment. Their advantage becomes more pronounced in the experiments with 200K target candidates, see Table 3. We suspect that this method is particularly robust when generalizing to words in non-parallel contexts: we find the same pattern in the BCWS task which is also constructed with nonparallel sentences.

Applying MIM brings consistent improvement for the best (avg_type) alignment method. Such improvements for the other methods are less stable. This suggests MIM is only effective when the alignment methods already learn a high-quality cross-lingual space before applying MIM.

As for training signals, relying only on a small dictionary (5K word pairs) yields comparable results with the methods that are trained on large amounts of parallel data. This suggests that a small seed dictionary may be enough to transfer the contextualized embeddings cross-lingually and be able to disambiguate words in context cross-lingually.

When comparing model variants, we see an advantage of aligning independent models over aligning shared models as we increase the training data. This advantage becomes more obvious with 200K target candidates, see Table 3. For EN–ES results in Figure 3, we observe that all alignment methods which use the shared model (i.e., multi_en→multi_es) start higher than results from aligning independently trained mono_en→multi_es. With the 'avg_type wa orig' method for example, aligning mono_en→multi_es starts at 29.04(%) whereas multi_en→multi_es starts at 34.07(%) given 100 parallel sentences. This is intuitive as English and Spanish share a larger portion of their vocabulary compared to English and Chinese: this gives the multilingual model a head start, but it is quickly surpassed by aligning from independently-trained models, especially via the avg_type alignment, as we increase training data.

In sum, we show that (1) BTSR is a challenging task; (2) unlike in Sentence Retrieval, context

	English original		token		avg_type		iso_type	
	mono_en	multi_en	wa mim	sa mim	wa mim	sa mim	wa mim	sa mim
mono_en→mono_zh	76.37	-	76.9	77.98	78.16	**78.28**	73.82	74.37
mono_en→multi_es	76.37	-	75.89	76.76	**77.2**	76.83	73.12	72.05
multi_en→multi_zh	-	72.6	73.56	**75.31**	74.89	75.1	68.55	68.07
multi_en→multi_es	-	72.6	72.3	73.78	**74.1**	73.72	68.43	66.99

Table 4: Evaluating alignment methods and model variants on the monolingual SCWS dataset which measures word similarity in context (in English). Spearman's ρ ($\times$ 100%). Previous best reported score is 69.3 (Neelakantan et al., 2014). Please refer to Table 1 for the explanations of the acronyms.

average type-level alignment performs the best in our task and in the BCWS task where the contexts are non-parallel, and can be further improved with the MIM technique. (3) Using a small dictionary is sufficient to transfer the contextualized embeddings via type-level alignment. (4) Aligning from a shared model gives a head start when two languages contain some shared vocabulary, but aligning from independently trained monolingual embeddings is able to achieve better performance given sufficient training data (5) Overall, increasing the search space from 20K to 200K target words results in a decrease of 10% in precision in BTSR, but the relative performance of different methods is more consistent and more pronounced.

Monolingual Contextual Evaluation. We also examine whether the cross-lingual alignment with MIM post-processing can improve the monolingual contextualized embeddings by evaluating the EN models on the Stanford Contextualized Word Similarity Task which measures similarity of word pairs with context in English. We evaluate the alignments learned from using 200K parallel sentences. The results are in Table 4. It seems that aligning independently trained models, which have better monolingual performance, outperforms aligning from shared models as found in BTSR. Also, we see consistent improvement over the original monolingual space after MIM, especially with avg_type alignment level. This indicates that the avg_type alignment level is effective not only in transferring the contextualized embeddings to the target language, but it can also improve the context-aware monolingual space.

We also observe that the EN contextualized models in their original space (both mono_en and multi_en) outperform SOTA (69.3%), a multi-sense static embedding model (Neelakantan et al., 2014). This indicates that the present contextualized embeddings are already capturing context effect including sense-level information without explicitly assigning embeddings to discrete sense categories.

6 Conclusion

We have conducted novel comparisons and analyses of various alignment methods for aligning contextualized embeddings cross-lingually. We have also introduced a novel task, Bilingual Token-level Sense Retrieval, which directly evaluates the retrieval of meaning-equivalent cross-lingual contextualized embeddings. The proposed task is challenging and enables a finer-grained analysis of different cross-lingual alignment methods. We have found that using context-average type-level alignment (avg_type) is effective and robust in transferring monolingual contextualized embeddings cross-lingually and at the same time improves the monolingual space. Using a small static dictionary as the alignment signal provides comparable results to word alignment methods relying on parallel corpora. We have also found that aligning independently trained monolingual embeddings yields better performance than aligning embeddings from a shared model. As our paper focuses only on the projection-based alignment methods, future work may explore other ways to learn the cross-lingual contextualized embeddings, e.g., based on joint training (Mulcaire et al., 2019).

Acknowledgments

We thank the anonymous reviewers for their helpful feedback on this work. We acknowledge Peterhouse College at University of Cambridge for funding Qianchu Liu's PhD research. The work was also supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909) awarded to Anna Korhonen. We also appreciate many helpful discussions and feedback from our colleagues in the Language Technology Lab, in particular Victor Prokhorov, Yi Zhu, Edoardo Ponti, Dr Haim Dubossarsky, and Dr Ehsan Shareghi.

References

Hanan Aldarmaki and Mona Diab. 2019. Context-aware cross-lingual mapping. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3906–3911, Minneapolis, Minnesota. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Francis Bond and Ryan Foster. 2013. Linking and extending an Open Multilingual WordNet. In *Proceedings of ACL*, pages 1352–1362.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. 2017. SemEval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26, Vancouver, Canada. Association for Computational Linguistics.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

Ta-Chung Chi and Yun-Nung Chen. 2018. CLUSE: Cross-lingual unsupervised sense embeddings. In *Proceedings of EMNLP*, pages 271–281.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yerai Doval, Jose Camacho-Collados, Luis Espinosa Anke, and Steven Schockaert. 2018. Improving cross-lingual word embeddings by meeting in the middle. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 294–304, Brussels, Belgium. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. In *Proceedings of ACL*, pages 710–721.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast bilingual distributed representations without word alignments. In *Proceedings of ICML*, pages 748–756.

Els Lefever and Veronique Hoste. 2009. SemEval-2010 task 3: Cross-lingual word sense disambiguation. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 82–87, Boulder, Colorado. Association for Computational Linguistics.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR, abs/1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeurIPS*, pages 3111–3119.

George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Tasnim Mohiuddin and Shafiq Joty. 2019. Revisiting adversarial autoencoder for unsupervised word translation with cycle consistency and improved training. In *Proceedings of NAACL-HLT*, pages 3857–3867.

Phoebe Mulcaire, Jungo Kasai, and Noah A. Smith. 2019. Polyglot contextual representations improve crosslingual transfer. In *Proceedings of NAACL-HLT*, pages 3912–3918.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per

word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proceedings of ACL*, pages 4996–5001.

Sebastian Ruder, Anders Søgaard, and Ivan Vulić. 2019. A survey of cross-lingual embedding models. *Journal of Artificial Intelligence Research*, 65:569–630.

Peter H Schönemann. 1966. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proceedigns of NAACL-HLT*, pages 1599–1613.

Ravi Sinha, Diana McCarthy, and Rada Mihalcea. 2009. SemEval-2010 task 2: Cross-lingual lexical substitution. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 76–81, Boulder, Colorado. Association for Computational Linguistics.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv preprint arXiv:1702.03859*.

Liang Tian, Derek F. Wong, Lidia S. Chao, Paulo Quaresma, Francisco Oliveira, Yi Lu, Shuo Li, Yiming Wang, and Longyue Wang. 2014. UM-corpus: A large English-Chinese parallel corpus for statistical machine translation. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1837–1842, Reykjavik, Iceland. European Language Resources Association (ELRA).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.

Ivan Vulić, Goran Glavaš, Roi Reichart, and Anna Korhonen. 2019. Do we really need fully unsupervised cross-lingual embeddings? In *Proceedings of EMNLP*.

Ivan Vulić and Marie-Francine Moens. 2013. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *Proceedings of NAACL-HLT*, pages 106–116.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011, Denver, Colorado. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Large-scale, Diverse, Paraphrastic Bitexts via Sampling and Clustering

J. Edward Hu **Abhinav Singh** **Nils Holzenberger**
Matt Post **Benjamin Van Durme**
Johns Hopkins University
{edward.hu,asingh78,nholzen1}@jhu.edu; {post,vandurme}@cs.jhu.edu

Abstract

Producing diverse paraphrases of a sentence is a challenging task. Natural paraphrase corpora are scarce and limited, while existing large-scale resources are automatically generated via back-translation and rely on beam search, which tends to lack diversity. We describe PARABANK 2, a new resource that contains multiple *diverse* sentential paraphrases, produced from a bilingual corpus using negative constraints, inference sampling, and clustering. We show that PARABANK 2 significantly surpasses prior work in both lexical and syntactic diversity while being meaning-preserving, as measured by human judgments and standardized metrics. Further, we illustrate how such paraphrastic resources may be used to refine contextualized encoders, leading to improvements in downstream tasks.

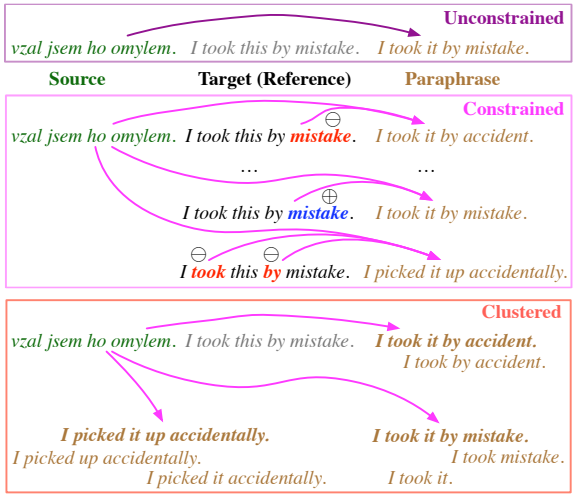


Figure 1: Contrived example paraphrases from previous work (unconstrained and constrained—used with permission) and ours (clustered).

1 Introduction

The ability to understand and produce paraphrases is a basic competency task, one that is often used as a teaching aid to validate if a student *understands* a statement or a concept. Current deep learning systems struggle with this task, exhibiting brittleness to both understanding and producing paraphrastic expressions (Iyyer et al., 2018).

One crucial factor behind this incompetence is the dearth of sentential paraphrastic data. Many works have sought to leverage the relative abundance of sub-sentential paraphrastic resources in paraphrase detection or generation (Napoles et al., 2016). Yet, they fail to capture contextualized word choices or syntactical variations, as word- or phrase-level resources cannot incorporate information from the whole input sentence.

Recent works have focused on leveraging bilingual resources to create large sentence-level paraphrastic collections using translation-based methods (Wieting and Gimpel, 2018; Hu et al., 2019). However, these works are confined to using beam search in decoding, which tend not to produce diverse candidates. One approach to force diverse translations is the use of hard lexical constraints at inference time (Hu et al., 2019). While effective in some cases, current approaches to automatic selection of such constraints is based on heuristics and task-oriented trial-and-error.

We present a novel resource with accurate and *collectively diverse* paraphrases, generated using stochastic decoding and clustering. By *collectively diverse*, we mean that the paraphrases of a given sentence cover a wide lexical and syntactic spectrum. Given a bilingual input pair, our core idea is to *sample* a large space of outputs from a translation system, cluster the results according to a notion of token-sequence similarity, score them with two translation models (one in each direction), and then select the best item from each cluster. We believe that sampling from the word distribution at each decoder time-step bet-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 44–54
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

ter preserves the decoder's level of uncertainty, which is intrinsic to the goals of paraphrasing. We also sample *ancillary* lexical constraints to discourage, instead of explicitly prohibiting (Hu et al., 2019), certain words from being used by the decoder. While our experiment produces a large-scale English resource, our approach is dependent only on the availability of large bitexts and so is language-agnostic. We chose to build an English resource from CzEng to enable a direct comparison with Wieting and Gimpel (2018) and Hu et al. (2019).

Our contributions include:

- A large, high quality paraphrase collection[1] with up to 5 paraphrases per reference, close to 100 million pairs in total, which are more diverse than prior work in two distinct ways, as measured by standardized metrics;
- An evaluation of semantic similarity, lexical and syntactic diversity, compared against prior works, along with results on Sentence Textual Similarity (STS) Benchmark;
- Experiments on how our resource can be leveraged to improve performance on a set of language tasks.

2 Paraphrase generation pipeline

Prior works in constructing sentential paraphrastic resources have worked from large collections of bitext, producing translations of the foreign language sentence which, when paired with the target-language reference, constitute a set of paraphrases. Working from the very large CzEng parallel corpus, Wieting and Gimpel (2018) produced a single paraphrase for each English sentence by translating from the Czech source. Hu et al. (2019) expanded on this by translating the Czech sentence several times, using positive or negative constraints obtained from the English reference.

In terms of producing diverse paraphrases, both approaches are limited because they rely on beam search. There are potentially billions of paraphrases of a sentence (Dreyer and Marcu, 2012), yet beam search with recurrent models can only search a constant subset of them (in the beam size). There are techniques for producing more diverse paraphrases, such as the use of positive and negative constraints (Hu et al., 2019) or syntactic fragments (Iyyer et al., 2018), but these require the user to manually specify them, which can be cumbersome and unreliable.

We follow these prior works in working with the CzEng, a Czech–English dataset (Bojar et al., 2016b), due to its size, diverse domain coverage, and rich syntactic variations (Wieting and Gimpel, 2018), and to allow for a direct comparison in methodologies. However, we propose a new approach to paraphrase generation designed to increase paraphrastic diversity, using a multi-step process: the first part of the pipeline generates a large number of candidate paraphrases through a random process, and the second part whittles them down to a much shorter list. For each {source, target} input pair, we run the following pipeline:

1. *Constrained sampling*. We sample translations using a source→target translation model with lexical constraints. We obtain negative constraints by randomly selecting a set of tokens from the "source", so that they are not allowed to appear in the translations. Then, we decode each translation by sampling from only the top-k most probable tokens at each time step, after excluding constrained tokens (§2.1).
2. *Dual scoring*. The set of samples is then scored against the original source input using a target→source translation model. The scores from the forward and backward models are summed (§2.2).
3. *Clustering*. The samples are then clustered. The best item from each cluster (according to the summed score) is then returned (§2.3).

2.1 Constrained sampling

Sampling is a more effective way to explore model search space than beam search, particularly in auto-regressive models that do not permit dynamic programming. We introduce two means by which we can expand the hypothesis space, and produce a more diverse set of paraphrases, relative to straightforward beam search.

Top-k sampling In auto-regressive neural MT, the standard sampling approach would be to choose a word w_t at each decoder timestep t by sampling from the distribution $P(w_t \mid w_{1\ldots t-1})$. This approach has been found effective over 1-best beam search in generating source sentences in

[1] Available at http://nlp.jhu.edu/parabank2

back-translation (Edunov et al., 2018). However, for paraphrasing, this is not ideal, since words that are not semantically licensed by the source may be selected. Instead, we propose top-k sampling, in which we choose w_t from the top k most-probable tokens at each time step. This way, we allow the model to sample flexibly, vastly opening up the hypothesis space, without creating a large risk of producing nonsensical translations.

Randomized negative constraints Negative constraints are tokens that are not permitted in the decoder output. They are not formally described in the literature, but an implementation was provided with the associated positive constraints (Post and Vilar, 2018). Negative constraints can be provided as tokens or phrases; the decoder tracks the progress of generation through each constraint and adds an infinite cost to the final word of any constraints, precluding its selection in both sampling and beam search.

In order to further increase sample diversity when generating the hypotheses (§2.1), we obtain negative constraints from the source by randomly choosing a subset of tokens. We do this independently multiple times for each input sentence. This provides new sets of constraints for the inputs, independent of the decoding.

Note that we use subword regularization (Kudo, 2018) during training, causing different subword segmentations to be applied to training data types each time they are encountered and helping to build more robust models. We only constrain on the Viterbi segmentation, effectively discouraging negatively constrained words from appearing in the output, instead of prohibiting them, since there are often ways for the model to produce a word by generating a different decomposition.

2.2 Back-translation likelihoods

Some semantic changes during paraphrasing, especially omission, are not well-reflected by the (forward) probability $p_{generate}$ from the generating model. However, a model running in the other direction can penalize this omission, as found by Goto and Tanaka (2017). Thus, we obtain the back-translation probability p_{back} of each sampled candidate paraphrase, and define the final score for each candidate paraphrase as the joint probability $p^* = p_{generate} * p_{back}$, which is the sum of negative log-likelihood.

2.3 Edit-distance-based clustering

The above process produces a large set of translations of the source sentence. Many of them will be minor variants of one another, but we expect that there will be a lot of variety in the large pool. The task now is to reduce this pool to a small set of *collectively diverse* paraphrastic candidates.

We address this problem with k-means clustering via Levenshtein (or edit) distance (Miller et al., 2009). We compute this on lowercased, segmented candidates, after striping punctuation. Clusters are initialized with the k furthest candidates measured by edit-distance. We also add the reference sentence as the centroid of an additional cluster and skip the re-centering for that cluster. This improves the chance of the k clusters congregating candidates different from the reference in different ways. When the clustering has converged, we take the candidate with the best score from each cluster (except for the one with the reference sentence), rank them by score, and take the best n as the final output.

3 Evaluations

3.1 Data

All of our experiments are based on the CzEng 1.7 corpus, a subset of CzEng 1.6 (Bojar et al., 2016b) that has been chosen for higher quality. Based on experience with data quality issues in neural MT (Ott et al., 2018; Junczys-Dowmunt, 2018), we decided to further clean the corpus. First, we normalize Unicode punctuation, and keep only bilingual pairs whose English side can be encoded with `latin-1` and Czech side with `latin-2`. We then filter the data with dual cross-entropy filtering (Junczys-Dowmunt, 2018). We use Sockeye (Hieber et al., 2017) to train two NMT models, CS–EN and EN-CS, on a relatively clean subset of the data provided for WMT 2018 (Bojar et al., 2016a): Europarl, Wiki titles, and news commentary. We use 4 layer Transformer models (Vaswani et al., 2017) trained to convergence, with held-out likelihood evaluated on a random 500-sentence subset of the WMT16 and WMT17 news test data. These models are then used to score all the remaining CzEng data after deduplication. We kept all sentences with a model score (negative log-likelihood) of less than 3.5. After applying the above two filters, we keep $19,723,003$ out of the $57,065,358$ pairs in CzEng 1.7.

3.2 Translation models

We train two new translation models on the filtered data, the CS–EN *generation model* (for generating English candidates via sampling) and the EN–CS *scoring model* (for providing backwards scores of the candidates). Both are Transformer models built with AWS SOCKEYE. The generation model is a 12 layer Transformer with a model and embedding size of 768, 12 attention heads, a feed-forward layer size of 3072. The scoring model has 6 layers, model and embedding size of 512, 8 attention heads, and a feed-forward layer size of 2048.

All training data is pre-processed with subword sampling using SentencePiece[2] (Kudo, 2018) with a vocabulary size of 20k and character coverage of 0.9999. We used separate models for Czech and English. At inference time, we use the Viterbi segmentation of each input sentence, for both the generation and scoring models.

3.3 Parameters

There are a few parameters involved in the sample-score-cluster pipeline. For each Czech input sentence, we generate 5 sets of random constraints (§2.1), creating 5 variants of the input. From each of these inputs, we generate 30 samples using top-k sampling with $k = 10$ (i.e., at each timestep, the model randomly chooses from the top 10 most probable words, according to their scaled distribution, and excluding negatively constrained words). The resulting 150 sentences are scored, and anything with a combined score greater than 3.5 is thrown out. The remaining sentences are clustered into 8 clusters, one of them centered on the English reference. The reference cluster is thrown out, and a list of the best-scoring translation from the remaining 7 clusters is constructed. From this list, the top 5 translations are returned as hypotheses.

3.4 Setup

We follow the evaluation framework of Hu et al. (2019), which judged semantic similarity between paraphrases and their reference through human evaluation, and lexical diversity via automatic metrics. We use the evaluation result made public by Hu et al. (2019) to enable a direct comparison. Rather than focusing on improving semantic similarity, which is limited by the quality of the bilingual resource, we seek to build a resource that contains both more lexical and syntactical diversity.

We obtained the evaluation set from Hu et al. (2019), which contains 400 English sentences from CzEng. Due to additional filtering, 24 out of 400 (6%) reference sentences aren't in PARABANK 2 and therefore excluded in this evaluation.

We set the output size $n = 5$. After sorting the candidates by negative log-likelihood for each reference, we treat candidates at each rank as an individual system to investigate the expected quality of paraphrases under our approach. For references that produce fewer than 5 paraphrases, the paraphrase with the highest negative log-likelihood is duplicated to fill in ranks that otherwise would be empty. We also artificially pick the paraphrase with the maximum, minimum, and median human semantic similarity judgment under each reference as three additional oracle systems.

3.5 Semantic similarity via human judgments

For a fair comparison, we used the evaluation setup released by Hu et al. (2019), which uses the interface from EASL (Sakaguchi and Van Durme, 2018) to collect semantic similarity and gammaticality judgments. Each human annotator is presented with a reference sentence and five paraphrases from different sources. Annotators use a slider bar under each paraphrase to rate the semantic similarity from 0 (Opposite/Irrelevant) to 100 (Identical Meaning). Annotators are also asked to comment on whether the paraphrase is ungrammatical or nonsensical. The reference sentence is repeated next to the paraphrase for easier visual comparison.

Each paraphrase receives at least 3 independent judgments. Following Hu et al. (2019), we randomly add in the reference sentence as a paraphrase and filter out annotators who fail to score them 100 more than 10% of such encounters. The result includes only annotators who contributed at least 25 judgments and is shown in Tab. 1.

3.6 Paraphrastic diversity

BLEU has been a successful metric in evaluating MT systems. However, as noted earlier, monolingual paraphrasing has inherently different objectives than cross-lingual translation. BLEU, in tandem with human evaluation in semantic similarity, makes a good metric for paraphrastic diversity.

[2] https://github.com/google/sentencepiece

System	Semantics↑	Grammar↑	1-BLEU↑	∩/∪ ↓	Tree ED↑	Len. Ratio
PARANMT	83.2	89.2	66.29	48.76	6.62	1.00
PARABANK$_{17}$	84.5	92.1	62.85	46.21	6.21	1.01
PARABANK$_{34}$	**85.7**	**92.7**	58.16	51.01	6.51	1.02
Our work$_1$	84.4±.0	90.2±.2	75.83±.10	37.75±.02	7.16±.05	1.04±.00
Our work$_2$	83.8±.0	88.3±.4	76.98±.07	36.19±.36	7.15±.17	1.05±.00
Our work$_3$	83.5±.0	87.3±.1	78.29±.69	35.22±.43	7.47±.11	1.05±.00
Our work$_4$	83.2±.2	86.6±.8	78.92±.19	34.49±.06	7.51±.11	1.06±.01
Our work$_5$	81.7±.1	87.3±.8	**81.55±.35**	**32.50±.32**	**7.80±.19**	1.09±.00
Our work$_{max}$	*91.2±.2**	*93.1±.8**	76.71±.11	37.15±.33	7.38±.06	1.05±.01
Our work$_{med.}$	84.1±.1	88.2±.1	78.34±.10	35.34±.25	7.52±.08	1.06±.00
Our work$_{min}$	72.5±.2	81.5±.2	*79.29±.21**	*33.13±.55**	*7.65±.10**	1.05±.00

Table 1: Paraphrastic diversity measured by (1-BLEU)×100, bag-of-word intersection/union score×100, and Tree edit-distance. Systems from this work that receive the best human judgments, worst human judgments, and the median, are included in the table. A higher 1-BLEU suggests higher paraphrastic diversity; a higher Intersection/Union score suggests a higher lexical diversity; a higher Tree edit-distance suggests a higher syntactic diversity. Best in each column, excluding oracle systems, is in bold. * denotes best oracle systems.

Here, we use 1-BLEU to measure how different the paraphrases are to the references.

We generate 5 paraphrases for each reference sentence using the approach outlined in this work. To account for randomness, we average over two independent runs in the result, shown in Tab. 1.

We consider two sources of paraphrastic diversity: 1) lexical diversity, the use of different words; and 2) syntactic diversity, the change of sentence or phrasal structure. We separately measure them using bag-of-word Intersection/Union scores and parse-tree edit-distances, respectively.

Lexical diversity A sentence is lexically different from the reference when it uses lexical paraphrases (e.g., synonyms) to convey similar meanings. We calculate the case-insensitive piece Intersection/Union score after striping punctuation and the SentencePiece white space symbol. All pieces are put to lowercase and into a set. The more pieces the two sentences share, the higher the score will be. The Intersection/Union scores between the reference and the paraphrases are shown in Tab. 1.

Syntactic diversity We consider the edit-distance between the parse trees of the reference and the paraphrase as a metric of syntactic diversity. Parse tree edit-distance is considered a useful feature in NLP tasks (Yao et al., 2013). The more syntactic variations there are between two sentences, the larger the tree edit-distance will be. We consider only the top 3 levels of the parse trees, excluding any terminals. Sentences are parsed with Stanford CoreNLP (Manning et al., 2014); the tree edit-distance is calculated with the APTED (Pawlik and Augsten, 2015a,b) algorithm. The average tree edit-distance for each system is shown in Tab. 1.

Diversity among paraphrases Hu et al. (2019) produced multiple paraphrases for each reference. While shown to be diverse compared to the reference, the authors did not investigate whether these paraphrases are trivial rewrites of one another, as it is likely the case with beam search under a few lexical constraints. Our clustering step is specifically designed to retrieve collectively diverse paraphrases.

We use the same metrics to evaluate pairs of systems from our work and compare them against PARABANK (Hu et al., 2019), as shown in Tab. 2. The max/min/median systems are oracle systems derived from human semantic similarity judgment scores. The human judgments from Tab. 1 show our paraphrases are of comparable quality to PARABANK, while maintaining a much higher degree of diversity among paraphrases of the same reference, as shown by automatic metrics.

3.7 Semantic similarity on STS Benchmark

In addition to evaluating via human judgments, we consider the same evaluation mechanism as PARANMT (Wieting and Gimpel, 2018): the use

Systems Compared	1-BLEU↑	∩/∪ ↓	Tree ED↑
PARABANK$_{17}$/PARABANK$_{34}$	20.58	80.93	2.26
Our work$_1$/Our work$_3$	64.16±.21	52.77±.48	5.51±.01
Our work$_3$/Our work$_5$	**71.05±.22**	**45.00±.51**	**6.40±.19**
Our work$_1$/Our work$_5$	69.46±.27	46.79±.12	6.25±.18
Our work$_{max}$/Our work$_{min}$	66.03±.86	49.10±.16	5.84±.33

Table 2: Collective diversity within our work compared to PARABANK, as measured by (1-BLEU)×100, intersection/union score×100, and parse tree edit-distance.

of paraphrase corpora as training data for the Semantic Textual Similarity (STS) task. STS aims to measure the degree of equivalence in meaning or semantics between a pair of sentences. Notably, Agirre et al. (2016) having been a part of the SemEval workshop (2012 -2017). The evaluation consists of human annotated English sentence pairs, scored on a scale of 0 to 5 to quantify similarity of meaning, with 0 being the least, and 5 the most similar.

Wieting and Gimpel (Wieting and Gimpel, 2018) compared three encoding mechanisms: WORD, TRIGRAM and LSTM. The WORD model (Wieting et al., 2016) averages the embedding for each word in the sentence into a fixed length vector embedding for the sentence; the TRIGRAM model (Huang et al., 2013) averages over character trigrams; and the LSTM (Hochreiter and Schmidhuber, 1997) approach averages over the final hidden states to obtain the sentence embedding.

Encoders are trained on paraphrase pairs (s, s') with a margin based loss function $l(s, s', t, t') =$

$$\max(0, \delta - \cos[g(s), g(s')] + \cos[g(s), g(t)]) + \max(0, \delta - \cos[g(s), g(s')] + \cos[g(s'), g(t')])$$

where g is one of (WORD, TRIGRAM, LSTM) and (t, t') is a negative sample selected from a *megabatch*, an aggregation of m mini-batches (Wieting and Gimpel, 2018).[3]

We evaluate the WORD model trained[4] on PARANMT, PARABANK and PARABANK 2 (our work). We retrieved the paraphrases from PARABANK and our work that share the same references as PARANMT-5M. Our work is evaluated as 5 systems, based on the rank in the output; the last available paraphrase is used when lower ranks are empty. We also include a system that uses a pair of paraphrases, instead of a reference and a paraphrase. We keep PARABANK paraphrases that have a bag-of-word intersection/union score of 0.7 or less, and use the 1-best based on regression scores. In Tab. 3, we report Pearson's r and Spearman's r on the STS'16 test set. Sentence embeddings trained on our work exhibit higher correlation with human judgments, which reflects the superior paraphrastic diversity of the corpus.

System	Pearson's r	Spearman's r
PARANMT	75.378	76.322
PARABANK	76.006	76.961
Our work$_1$	**76.546**	77.528
Our work$_2$	76.143	77.240
Our work$_3$	76.397	77.500
Our work$_4$	76.414	**77.612**
Our work$_5$	75.882	77.075
Our work$_{1/5}$	75.680	76.882

Table 3: Pearson's r × 100 and Spearman's r × 100 computed on STS 2016 task. Our work$_{1/5}$ contains paraphrase pairs from system$_1$ paired with system$_5$, while all other systems are paired with the reference sentence.

3.8 Improving contextualized encoders with paraphrastic data

Paraphrastic data can be used to fine-tune contextualized encoders such as BERT (Devlin et al., 2018). We frame the fine-tuning task as paraphrase identification (Das and Smith, 2009), where given a pair of sentences, the task is to classify them as paraphrases or non-paraphrases. To generate the training data, we extract, for each

[3] We confirmed this loss with Wieting and Gimpel, that it captures their open implementation, which we employ. Wieting and Gimpel (2018) described their loss as: $\max(0, \delta - \cos(g(s), g(s')) + \cos(g(s), g(t)))$, which is equivalent under their assumption the paraphrases are equivalent.

[4] https://github.com/jwieting/para-nmt-50m

	QQP	MNLI	STS-B	MRPC
BERT	87.90	**83.86**	88.40	84.00
pBERT	**88.14**	82.64	**88.59**	**86.55**

Table 4: F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for MNLI. Numbers reported on Dev set

	Type	BERT	pBERT
F1	HasAns	**76.81**	74.21
	NoAns	71.44	**74.95**
	Total	74.12	**74.58**
Exact Match	HasAns	**70.34**	68.00
	NoAns	71.44	**74.95**
	Total	70.89	**71.48**

Table 5: SQuAD 2.0 results on dev set.

sentence in PARANMT-5M, the sentence embeddings generated by the WORD model trained in §3.7. For each sentence s, we then find the (approximate) nearest neighbour n which is not s', among all of the sentences. We thus obtain two pairs, where (s, s') is a paraphrase pair, and (s, n) is a non-paraphrase pair. We use these to train a binary classifier with cross-entropy loss.

We then use this BERT fine-tuned on paraphrases (henceforth pBERT) for fine-tuning on SQuAD 2.0 (Rajpurkar et al., 2018) and 4 NLP tasks present in the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019): Quora Question Pairs (QQP) (Chen et al., 2017), Multi-Genre Natural Language Inference (MNLI) (Williams et al., 2018), the Semantic Textual Similarity Benchmark (STS-B) (Agirre et al., 2016), and the Microsoft Research Paraphrase Corpus (MRPC) (Dolan et al., 2004). Following the model formulation, hyper-parameter selection and training procedure specified in Devlin et al. (2018), we add a single task-specific, randomly initialized output layer for the classifier.

We present our results in Tab. 4 and Tab. 5. We observe gains for STS-B, MRPC and QQP, tasks strongly related to paraphrase identification. Fine-tuning on our paraphrase corpus also improves performance on SQuAD, a question-answering task, while slightly degrading performance on MNLI. Overall, simple fine-tuning of BERT on our corpus leads to improvements on downstream tasks, in particular when the task is related to paraphrase detection.

4 Related works

4.1 Paraphrastic resources

Paraphrastic resources exist across different scopes (i.e., lexical, phrasal, sentential) and different creation strategies (i.e., manually curated, automatically generated). For a more comprehensive survey on data-driven approaches to paraphrasing, please refer to Madnani and Dorr (2010).

Sub-sentential resources WordNet (Miller, 1995), FrameNet (Baker et al., 1998), and VerbNet (Schuler, 2006) can be used to extract paraphrastic expressions at lexical levels. They contain the grouping of words or phrases that share similar semantics and sometimes entailment relations. While FrameNet and VerbNet do have example sentences or frames where lexical units are put into contexts, there is no explicit paraphrastic relations among these examples. Also, these datasets tend to be small, as they were curated manually. There have been efforts to augment such resources with automatic methods (Snow et al., 2006; Pavlick et al., 2015b), but they are still confined to lexical level and sometimes require the use of other paraphrastic resources (Pavlick et al., 2015b).

PPDB (Ganitkevitch et al., 2013; Pavlick et al., 2015a) automated the generation of lexical paraphrases via bilingual pivoting, taking advantage of the relative abundance of bilingual corpora. While significantly larger and more informative (e.g., ranking, entailment relations, etc.) than the above manually curated resources, PPDB suffers from ambiguity as words or phrases are removed from their sentential contexts.

Sentential resources There exists multiple human translations in the same language for some classic readings. Barzilay and McKeown (2001) sought to extract lexical paraphrastic expression from such sources. Unfortunately such resources – along with those manually constructed for text generation research (Robin, 1995; Pang et al., 2003) – are small and limited in domain.

PARANMT and PARABANK are two much larger sentential paraphrastic resources created through back-translation.

Reference:		
	Real life is sometimes thoughtless and mean.	Hey, stop right there!
PARANMT:		
	real life is sometimes reckless and cruel .	hey , stop .
PARABANK:		
	The real life is occasionally ruthless and cruel.	Stay where you are!
	The real world is occasionally ruthless and cruel.	
	The real life is sometimes reckless and cruel.	
Our work:		
	True life is sometimes ruthless and cruel.	Hold your position!
	Actual life is sometimes ruthless and cruel.	Stay where you are!
	Sometimes real life is ruthless and cruel.	Stay in position!
	Real life can be inconsiderate, cruel sometimes.	Remain where you are!
	Real living is a harsh and unscrupulous one, at times.	Stay put!

Table 6: Selected examples from our work, compared to paraphrastic resources with prior approaches. Our work has paraphrases that are not only different from the reference, but also diverse among themselves.

4.2 Translation-based Approaches

PARANMT is an automatically generated sentential paraphrastic resource through back-translating bilingual resources. It leveraged the imperfect ability of Neural Machine Translation (NMT) to recreate the translation target by conditioning on the source side of the bitext.

PARABANK took a similar approach but with the inclusion of lexical constraints from the target side of the bitext. This step allows for multiple translations from one bilingual sentence pair and promotes lexical diversity. Their work, despite being larger and shown to be less noisy than PARANMT, relies on heuristics to produce *hard* constraints on the decoder, which often causes unintended changes in semantics or grammar.

Both works largely follow standard approaches in NMT, generating 1-best hypotheses given a source text and a set of constraints using beam search. Sentential paraphrasing, nevertheless, has fundamentally different objectives than MT. The latter strives to find the best elicitation that is both fluent and semantically close to the **foreign** text to convey information across languages. The former, on the other hand, seeks syntactically and lexically diverse expressions that convey the same meaning, with the goal of capturing the **intrinsic flexibility and uncertainty** of human communications. This work attempts to adapt the methodology to these objectives of monolingual paraphrasing.

4.3 Leveraging paraphrases in NLP

In the context of semantic parsing, Berant and Liang (2014) use a paraphrase classification module to determine the match between a canonical utterance and a logical form, both using a phrase table and distributed representations. To improve question answering (QA), Duboue and Chu-Carroll (2006) generate paraphrases of a given question using back-translation, and optionally replace the original question with the most relevant paraphrase. Dong et al. (2017) tackle QA by marginalizing the probability of an answer over a set of paraphrases, generated using rule-based and NMT-based methods. Fader et al. (2013) use a corpus of questions with paraphrases, to construct a corpus of semantically equivalent queries.

The task of paraphrase identification, which we use as a fine-tuning objective, has been studied as a task in itself. Das and Smith (2009) use grammars to perform generative modeling of paraphrases. Madnani et al. (2012) identify paraphrases by relying only on MT metrics as features. Ferreira et al. (2018) feed sentence similarity measured with hand-crafted features to machine learning algorithms. Convolutional neural networks have been introduced by Yin and Schütze (2015) and Chen et al. (2018), and further augmented with LSTMs (Kubal and Nimkar, 2018) and attention mechanisms (Fan et al., 2018).

5 Conclusions and future work

A presumed goal for building a sentential paraphrase resource is to capture *different* ways of expressing the same thing: *diversity matters*. Previous work on paraphrastic resource creation relied on decoding techniques from NMT using bilingual corpora, with limited success in promoting diverse expressions. We have presented a new community resource produced by sampling and clustering. We evaluated our method against prior works (Wieting and Gimpel, 2018; Hu et al., 2019) and found significant gains in both lexical and syntactic diversity. Further, we've shown how straightforward fine-tuning of a state-of-the-art contextual encoder on our resource can improve performance on a variety of language tasks.

Acknowledgments

This work was supported in part by a National Science Foundation collaborative grant (BCS-1748969/BCS-1749025) The MegaAttitude Project: Investigating selection and polysemy at the scale of the lexicon.

References

Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval@NAACL-HLT*, pages 497–511. The Association for Computer Linguistics.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of ACL/ICCL*, ACL '98, pages 86–90, Stroudsburg, PA, USA.

Regina Barzilay and Kathleen R McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1415–1425.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016a. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, volume 2, pages 131–198.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. 2016b. CzEng 1.6: Enlarged Czech-English Parallel Corpus with Processing Tools Dockered. In *Text, Speech, and Dialogue: 19th International Conference, TSD 2016*, number 9924 in Lecture Notes in Computer Science, pages 231–238, Cham / Heidelberg / New York / Dordrecht / London. Masaryk University, Springer International Publishing.

Peixin Chen, Wu Guo, Zhi Chen, Jian Sun, and Lanhua You. 2018. Gated convolutional neural network for sentence matching. *memory*, 1:3.

Zihan Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2017. First quora dataset release: Question pairs.

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 468–476. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. *arXiv preprint arXiv:1708.06022*.

Markus Dreyer and Daniel Marcu. 2012. Hyter: Meaning-equivalent semantics for translation evaluation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 162–171. Association for Computational Linguistics.

Pablo Duboue and Jennifer Chu-Carroll. 2006. Answering the question you wish they had asked: The impact of paraphrasing for question answering. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1608–1618.

Miao Fan, Wutao Lin, Yue Feng, Mingming Sun, and Ping Li. 2018. A globalization-semantic matching neural network for paraphrase identification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2067–2075. ACM.

Rafael Ferreira, George DC Cavalcanti, Fred Freitas, Rafael Dueire Lins, Steven J Simske, and Marcelo Riss. 2018. Combining sentence similarities measures to identify paraphrases. *Computer Speech & Language*, 47:59–73.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings NAACL-HLT 2013*, pages 758–764.

Isao Goto and Hideki Tanaka. 2017. Detecting untranslated content for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 47–55, Vancouver. Association for Computational Linguistics.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. *CoRR*, abs/1712.05690.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

J. Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. 2019. PARABANK: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. In *Proceedings of AAAI 2019*, Hawaii, USA. AAAI.

Po-Sen Huang, , Jianfeng Gao, , and and. 2013. Learning deep structured semantic models for web search using clickthrough data. ACM International Conference on Information and Knowledge Management (CIKM).

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Marcin Junczys-Dowmunt. 2018. Dual conditional cross-entropy filtering of noisy parallel corpora. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 888–895. Association for Computational Linguistics.

Divesh R Kubal and Anant V Nimkar. 2018. A hybrid deep learning architecture for paraphrase identification. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75. Association for Computational Linguistics.

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Frederic P. Miller, Agnes F. Vandome, and John McBrewster. 2009. *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau?Levenshtein Distance, Spell Checker, Hamming Distance*. Alpha Press.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Courtney Napoles, Chris Callison-Burch, and Matt Post. 2016. Sentential paraphrasing as black-box machine translation. In *Proceedings of the NAACL 2016*, pages 62–66, San Diego, California. Association for Computational Linguistics.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3956–3965, Stockholmsmssan, Stockholm Sweden. PMLR.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of HLT/NAACL*.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015a. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of ACL/IJCNLP*, volume 2.

Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015b. Framenet+: Fast paraphrastic tripling of framenet. In *Proceedings of the ACL/IJCNLP*, volume 2.

Mateusz Pawlik and Nikolaus Augsten. 2015a. Efficient computation of the tree edit distance. *ACM Trans. Database Syst.*, 40(1):3:1–3:40.

Mateusz Pawlik and Nikolaus Augsten. 2015b. Tree edit distance: Robust and memory-efficient. *Information Systems*, 56.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822.

Jacques Pierre Robin. 1995. *Revision-based Generation of Natural Language Summaries Providing Historical Background: Corpus-based Analysis, Design, Implementation and Evaluation*. Ph.D. thesis, New York, NY, USA. UMI Order No. GAX95-33653.

Keisuke Sakaguchi and Benjamin Van Durme. 2018. Efficient online scalar annotation with bounded support. In *Proceedings of ACL*, pages 208–218, Melbourne, Australia. ACL.

Karin Kipper Schuler. 2006. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of ICCL/ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198.

John Wieting and Kevin Gimpel. 2018. PARANMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of ACL 2018*, pages 451–462. ACL.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 858–867, Atlanta, Georgia. Association for Computational Linguistics.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Large-scale representation learning from visually grounded untranscribed speech

Gabriel Ilharco†* **Yuan Zhang**‡ **Jason Baldridge**‡
†Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA, USA
‡Google Research, Mountain View, CA, USA
gamaga@cs.washington.edu, {zhangyua,jridge}@google.com

Abstract

Systems that can associate images with their spoken audio captions are an important step towards visually grounded language learning. We describe a scalable method to automatically generate diverse audio for image captioning datasets. This supports pretraining deep networks for encoding both audio and images, which we do via a dual encoder that learns to align latent representations from both modalities. We show that a masked margin softmax loss for such models is superior to the standard triplet loss. We fine-tune these models on the Flickr8k Audio Captions Corpus and obtain state-of-the-art results—improving recall in the top 10 from 29.6% to 49.5%. We also obtain human ratings on retrieval outputs to better assess the impact of incidentally matching image-caption pairs that were not associated in the data, finding that automatic evaluation substantially underestimates the quality of the retrieved results.

1 Introduction

Natural language learning in people starts with speech, not text. Text is tidy: it comes in convenient symbolic units that vary little from one writer to another. Speech is continuous and messy: the sounds used to convey a given word are modified by those of surrounding words, and the rate of speech, its pitch, and more vary across speakers and even for the same speaker in different contexts. As such, problems involving speech provide distinct challenges and opportunities for learning language representations that text-based work—which represents the vast majority—gets a free pass on.

Recent work has explored various means to transform raw speech into symbolic forms with little or no supervision (Park and Glass, 2007; Varadarajan et al., 2008; Ondel et al., 2016; Kamper et al., 2017a; Bhati et al., 2018). However, learning natural language starts with grounded, contextualized speech. While infants as young as 8-months-old can segment word-like units without non-linguistic information (Jusczyk and Aslin, 1995) and adults can learn to segment words in artificial languages (Saffran et al., 1996), a learner must ultimately ground their representations of linguistic sequences (Harnad, 1990) to effectively use them to refer to objects, events and more. Furthermore, learning from rich perceptual data and interactions can be more efficient as it provides additional cues to the identities of words and their meaning in context.

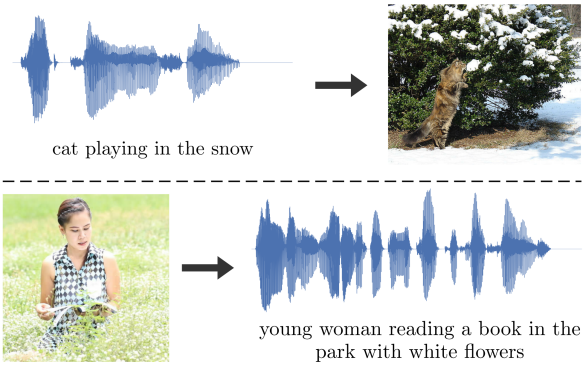


Figure 1: Models that encode speech segments and images into a shared latent space enable images to be retrieved using their audio descriptions (top) and to associate images with spoken captions (bottom). Text captions are provided for clarity; only speech and images are used by the models.

We address the problem of relating images to audio captions that describe them (Figure 1), building on previous research into learning from visually grounded, untranscribed speech (Harwath and Glass, 2015; Sun et al., 2016; Harwath et al., 2016; Chrupała et al., 2017; Kamper et al., 2017b; Chrupała, 2019; Harwath and Glass, 2019). Such problem settings provide opportunities both to improve our theoretical understanding of language

* Work done as a member of the Google AI Residency Program.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 55–65
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

as well as to realize gains on practical problems—including voice interaction with virtual assistants, image retrieval based on speech, and generally better supporting people with visual impairments.

Our contribution is to improve performance on bidirectional speech/image retrieval through better data and better models for learning fixed dimensional latent representations of both modalities. We construct a synthetic speech caption dataset for pretraining by applying text-to-speech (TTS) on Conceptual Captions (Sharma et al., 2018), a dataset with 3.3 million diverse image-caption pairs. Unlike Chrupała et al. (2017), who similarly applied TTS to MS-COCO (Chen et al., 2015), we inject diversity by varying the voice, speech rate, pitch and volume gain on every synthetically produced audio caption. We refer to the resulting dataset as Conceptual Spoken Captions (CSC). CSC's scale allows us to train deeper models than previous work. We use Inception-ResNet-v2 (Szegedy et al., 2017) to encode both the audio and visual modalities in a dual encoder model, pretraining on CSC and then fine-tuning and evaluating on human speech in the smaller Flickr Audio Caption Corpus (FACC) (Harwath and Glass, 2015). Using an adapted batch loss function rather than the triplet loss used in previous work, we substantially improve on the previous state-of-the-art for the standard FACC retrieval tasks.

Image captioning datasets contain positively paired items—but that does not imply that a random image and caption cannot also be a valid match. For instance, in FACC there are many spoken captions about beaches and sunsets and plenty of images that match these captions; two different images with descriptions *"A surfer is riding a wave."* and *"A man surfs the wave"* are likely compatible. It is of course not feasible to exhaustively annotate all pairwise associations, so we have human raters judge the top five retrieved results for two models to assess the impact of this aspect of the data on automatic retrieval metrics used thus far. Unsurprisingly, models retrieve many compatible results that are unpaired in FACC: with the human evaluations, we find consistent increases in recall.

2 Data

Larger training datasets support better performance and generalization (Banko and Brill, 2001; Halevy et al., 2009; Sun et al., 2017), especially for deep models. Collecting labels from people has become easier via crowd computing (Buhrmester et al., 2011), but is still expensive and remains a bottleneck for creating broad and representative datasets. This motivates the case for exploiting incidental annotation (Roth, 2017) and automating some aspects of dataset creation. The current trend of using machine translation systems to produce augmented datasets for machine translation itself (Sennrich et al., 2016) and for monolingual tasks like classification (Yu et al., 2018) and paraphrasing (Wieting and Gimpel, 2018) is a good example of this.

For speech image captioning, Chrupała et al. (2017) used a Text-to-Speech (TTS) system to create audio from the textual captions given in the MS-COCO dataset, resulting in 300k unique images with 5 spoken captions each. We scale this idea to the larger and more diverse textual Conceptual Captions dataset with 3.3 million unique image and captions, additionally modifying the produced speech by using multiple voices and random perturbations to the rate, pitch and audio. Our goal is to make the resulting data more effective for pretraining models so they can learn more efficiently on smaller amounts of human speech.

2.1 Conceptual Captions

Image captioning datasets have ignited a great deal of research at the intersection of the computer vision and natural language processing communities (Lin et al., 2014; Vinyals et al., 2015; Bernardi et al., 2016; Anderson et al., 2018). Getting annotators to provide captions works well with crowd computing, but Sharma et al. (2018) exploit incidental supervision for this task to obtain greater scale with their Conceptual Captions dataset. It contains 3.3 million pairs of image and textual captions, where pairs are extracted from HTML web pages using the *alt-text* field of images as a starting point for their descriptions.

The textual captions are processed in a hypernymization stage. Named entities and syntactic dependency annotations are obtained using Google Cloud Natural Language APIs, which are matched to hypernym terms using the Google Knowledge Graph Search API. Proper nouns, numbers, units, dates, durations and locations are removed; identified named-entities are substituted with their hypernym, merging together analogous terms when possible. For example, the original alt-text (1) is converted to the conceptual caption (2).

(1) **alt-text**: *Musician Justin Timberlake per-*

forms at the 2017 Pilgrimage Music & Cultural Festival on September 23, 2017 in Franklin, Tennessee.

(2) **conceptual caption**: *pop artist performs at the festival in a city.*

There are many sequential filtering steps for improving the quality of the captions—see Sharma et al. (2018) for a thorough description. As quality control, a random sample of 4K conceptual captions were rated by human annotators, and 90.3% were judged "good" by at least 2 out of 3 raters.

2.2 Conceptual Spoken Captions

We use TTS to generate a high-fidelity spoken sentence for each of the 3.3 million textual captions in the Conceptual Captions dataset.[1] We use the Google Cloud Speech API[2] for TTS. Internally, the service uses a WaveNet model (Van Den Oord et al., 2016) to generate audio. For diversity, the speech is synthesized using parameter variations, as follows:

- *Voice*, which is sampled uniformly from a set of 6 different voices generated using a WaveNet model for American English.
- *Speaking rate* controls the speed of the synthesized audio. A speaking rate of 1.0 means the normal speed of a given voice, while a speaking rate of 2.0 means twice as fast. When synthesizing the data, we draw this parameter from a Gaussian distribution $\sim \mathcal{N}(1.0, 0.1^2)$.
- *Pitch* controls how high/deep the voice is. For example, if set to 1, this means the voice will be synthesized 1 semitones above the original pitch. This parameter is drawn from a Gaussian distribution $\sim \mathcal{N}(0.0, 1.0^2)$.
- *Volume gain* controls a gain in dB with respect to the normal native signal amplitude. If set to 0, the voice is synthesized without alterations in volume. This parameter is drawn from a Gaussian distribution $\sim \mathcal{N}(0.0, 2.0^2)$.

To avoid degenerate cases, we clip the values sampled from the Gaussian distributions described above such that they are never more than 2 times the standard deviation from the mean. All spoken captions are generated in 16000 Hz.

[1]The alt-text does not come with the dataset and cannot be redistributed, so we focus on the conceptual captions for ease of experimentation and reproducibility.

[2]https://cloud.google.com/text-to-speech/

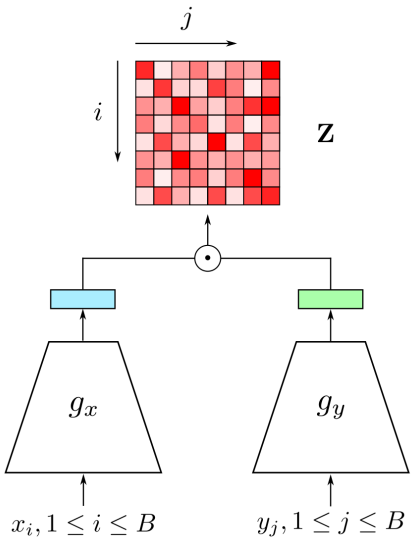


Figure 2: Dual-encoder model architecture.

2.3 Flickr Audio Caption Corpus

The Flickr Audio Caption Corpus (FACC) (Harwath and Glass, 2015) consists of 40,000 pairs of images and spoken captions, with 8000 unique images, of which 1000 are held for validation and 1000 for testing. The spoken captions are generated from humans reading the textual captions from the Flickr8k dataset (Hodosh et al., 2013), originally crowd-sourced and based on images from Flickr.

We use FACC for evaluation, both when pretraining on Conceptual Spoken Captions and when training on FACC from scratch. Like previous work, the core evaluation considered is retrieval of the known paired image given an audio caption within some top-k set of retrieved items (e.g. R@1 for whether the first item retrieved is the paired one and R@10 for whether it is in the top ten results). We also conduct human evaluations on retrieval outputs to detect the presence of unpaired but matching image-caption pairs identified by the models and thereby better assess their impact on performance.

3 Model

Dual encoders are used in a wide range of applications, including signature verification (Bromley et al., 1994), object tracking (Bertinetto et al., 2016), sentence similarity (Mueller and Thyagarajan, 2016), improving neural machine translation (Yang et al., 2019) and many others. The core of this set of architectures is a simple two-tower model illustrated in Figure 2, where inputs $x \in \mathcal{X}$ are processed by an encoder g_x and inputs $y \in \mathcal{Y}$ by a second encoder g_y. The inputs may come

from the same distribution—or they may be from entirely different sources or modalities. The towers may share the same architecture and weights—or they can be completely unlike and disconnected.

These models are standard in audiovisual image captioning (Harwath and Glass, 2015; Chrupała, 2019; Harwath et al., 2018). In this setting, the dual encoder model, is composed by a visual tower, g_{vis}, processing the images, and an audio tower, g_{aud}, processing the spoken captions. The model is trained to map both modalities into a joint latent space. Here, we extend previous work to consider a batched margin loss, which we show to be superior for learning dense representations for retrieval.

Notation. The inputs are processed in batches of size B. For each input x_k and y_k in the batch, $1 \leq k \leq B$, let $g_x(x_k)$ and $g_y(y_k)$ be their latent representations extracted by the corresponding tower. We define the $B \times B$ matrix $\mathbf{Z}$ as the similarity between the latent representations for each pair of elements in the batch. A natural choice for that similarity is the dot product between the latent representations:

$$\mathbf{Z}_{ij} = g_x(x_i) \cdot g_y(y_j) \tag{1}$$

As shown in Figure 2, $\mathbf{Z}$ encodes all pairwise associations in the batch. However, an additional aspect of some datasets must be taken into account: often times the same input x can match multiple inputs y or vice-versa—for instance, both Flickr8k and MS-COCO have multiple captions for the each image. To respect these pairs when they land in the same batch—and thus not penalize models for (correctly) associating them—we define a $B \times B$ masking matrix $\mathbf{M}$:

$$\mathbf{M}_{ij} = \begin{cases} 0, & \text{if } x_i \text{ matches } y_j \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

All pairs (x_k, y_k) match and this equivalence is transitive, so $\mathbf{M}$ is symmetric and all diagonal elements $\mathbf{M}_{kk}$, $1 \leq k \leq B$ are zero.

Triplet Loss. Both Chrupała (2019) and Harwath et al. (2018) (and their previous work) employ the triplet loss function given in Equation 3.

$$\mathcal{L}_{\text{T}} = \sum_{k=1}^{B} \big(\max(0, \mathbf{Z}_{km} - \mathbf{Z}_{kk} + \delta) + \max(0, \mathbf{Z}_{nk} - \mathbf{Z}_{kk} + \delta) \big) \tag{3}$$

For each value k, m is randomly drawn from a uniform distribution over indices j such that $M_{kj} = 1$, and n over indices i such that $M_{ik} = 1$.

Masked Margin Softmax Loss. The triplet loss (3) used previously misses opportunities to learn against a wider set of negative examples, namely all those in the batch that are not known to be positively associated (i.e., $\mathbf{M}_{ij} = 1$). To exploit these additional negatives, we minimize the Masked Margin Softmax (MMS) loss function, inspired by Henderson et al. (2017) and Yang et al. (2019). MMS simulates x-to-y and y-to-x retrievals inside the batch. It is defined at a high level as:

$$\mathcal{L}_{\text{MMS}} = \mathcal{L}_{xy} + \mathcal{L}_{yx} \tag{4}$$

$\mathcal{L}_{\text{MMS}}$ is the sum of losses defined over x-to-y (Eq. 5) and y-to-x (Eq. 6) in-batch retrievals.

$$\mathcal{L}_{xy} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{e^{\mathbf{Z}_{ii} - \delta}}{e^{\mathbf{Z}_{ii} - \delta} + \sum_{j=1}^{B} \mathbf{M}_{ij} e^{\mathbf{Z}_{ij}}} \tag{5}$$

$$\mathcal{L}_{yx} = -\frac{1}{B} \sum_{j=1}^{B} \log \frac{e^{\mathbf{Z}_{jj} - \delta}}{e^{\mathbf{Z}_{jj} - \delta} + \sum_{i=1}^{B} \mathbf{M}_{ij} e^{\mathbf{Z}_{ij}}} \tag{6}$$

These are equivalent to a cross-entropy loss after a column-wise or row-wise softmax on the matrix $\mathbf{Z}$, subject to the masking constraints in $\mathbf{M}$ and margin δ.

The margin hyperparameter δ is gradually increased as training progresses. Empirically, we found that, with a fixed δ, large values lead to unstable performance in early training, while small values lead to negligible results in final performance. Starting with a small δ and increasing it does not hurt early training and forces the model to learn from a harder task later on. There many ways to increase δ along training—e.g. linearly, quadratically, and exponentially. The latter is used in this work.

Contrasting Equations 3 and 4, the former chooses a negative sample randomly, while the latter takes advantage of all negative pairs in the batch and thus improves sample efficiency. $\mathcal{L}_{\text{MMS}}$ has three main differences with Yang et al. (2019): (1) a masking term that accounts for the fact that there might be multiple positive choices in the batch for a given input; (2) a varying margin term δ, which is increased during training; (3) a log term that makes MMS more closely resemble a cross-entropy loss.

		Speech to Image					Image to Speech				
Loss	Batch Size	R@1	R@5	R@10	R@50	R@100	R@1	R@5	R@10	R@50	R@100
$\mathcal{L}_{\text{T}}$	48	.037	.109	.165	.367	.474	.031	.101	.155	.346	.455
$\mathcal{L}_{\text{MMS}}$	12	.025	.083	.129	.311	.432	.024	.083	.132	.315	.433
	24	.054	.143	.206	.418	.533	.046	.137	.197	.411	.520
	48	**.078**	**.204**	**.282**	**.499**	**.604**	**.074**	**.194**	**.269**	**.485**	**.587**

Table 1: Performance on the validation set of Conceptual Spoken Captions, comparing different loss functions and batch sizes.

4 Experiments

4.1 Experimental settings

Image preprocessing. During training, data augmentation is performed by randomly distorting the brightness and saturation of images. Each image is also randomly cropped or padded such that at least 67% of the area of the original image is covered, and re-scaled if necessary to 299×299. During evaluation, we do not perform color distortions, and we crop/pad the central portion of the images.

Audio preprocessing. We extract 128 Mel-Frequency Cepstral Coefficients (MFCCs) from the raw audio signals using a window size of 20ms. The audio signals have a sampling rate of 16000Hz. We compute features every 10ms, such that each window has a 50% overlap with its neighbors. During training, we randomly crop/pad the MFCCs in the temporal dimension, and perform data augmentation as in Park et al. (2019), using one mask with a frequency mask parameter of 20 and a time mask parameter of 40. We do not perform time warping.

Encoders. Both audio and image encoders are Inception-ResNet-v2 networks (Szegedy et al., 2017), allowing the model to reap the benefits of relatively low computational cost, fast training and and strong performance when combining the Inception architecture with residual connections.[3] Related to our setting for audio processing, Li et al. (2019) also uses residual convolutional neural networks for state of the art results on LibriSpeech dataset (Panayotov et al., 2015). For the audio tower, we stack 3 replicas of the MFCCs and treat them as images. For each modality, a 1536-dimensional latent space representation is extracted. Despite using the same architecture for both encoders, their weights are not shared. Unless specified otherwise, the models are *not* pretrained.

[3] See Bianco et al. (2018) for an extensive benchmark analysis of popular convolutional neural network architectures.

Optimization. Models are trained using Adam (Kingma and Ba, 2014), with an initial learning rate of 0.001 and an exponential decay of 0.999 every 1000 training steps, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1\text{e}{-8}$. We use a weight decay of $4\text{e}{-5}$, and train on 32 GPUs until convergence. Unless specified otherwise, the optimization objective is minimizing the loss $\mathcal{L}_{\text{MMS}}$ (Eq. 4) with a margin term initially set to $\delta = 0.001$ exponentially and increased by a factor of 1.002 every 1000 steps.

4.2 Retrieval: Conceptual Spoken Captions

Our primary aim with CSC is to use it for pretraining for later fine-tuning and evaluation on datasets with human speech instead of TTS. Nevertheless, we can compare different loss functions and different batch sizes on the CSC validation set to better understand the impact of these parameters.

We train models on CSC for 3 million steps, cropping/padding spoken captions to a duration of 3.5 seconds and using the loss functions $\mathcal{L}_{\text{T}}$ (Eq. 3) and $\mathcal{L}_{\text{MMS}}$ (Eq. 4). We find continuing improvements as batch size increases from 12 to 24 to 48. Furthermore, with the same batch size of 48, models optimized for minimizing $\mathcal{L}_{\text{MMS}}$ perform substantially better than those using $\mathcal{L}_{\text{T}}$, as summarized in Table 1. Of particular note is that R@1 scores for $\mathcal{L}_{\text{MMS}}$ (batch size 48) are more than double those of $\mathcal{L}_{\text{T}}$ in both directions.

4.3 Retrieval: Flickr Audio Caption Corpus

Table 2 compares previous results on the FACC dataset with those obtained by variations of our model. As a pre-processing step, spoken captions are cropped/padded to a duration of 8 seconds. After pretraining the model in CSC, we explore all possible combinations of using or not the pretrained weights for each of the branches g_{aud} and g_{vis} as a warm-starting point, training until convergence on FACC. Warm-starting each of the branches in the dual-encoder leads to substantial improvements

	Model	Caption to Image					Image to Caption				
		R@1	R@5	R@10	R@50	R@100	R@1	R@5	R@10	R@50	R@100
Text	Socher et al. 2014	-	-	.286	-	-	-	-	.2r90	-	-
	Karpathy et al. 2014	-	-	.425	-	-	-	-	.440	-	-
	Harwath and Glass 2015	-	-	.490	-	-	-	-	**.567**	-	-
	Chrupała et al. 2017	**.127**	**.364**	**.494**	-	-	-	-	-	-	-
Speech	Harwath and Glass 2015	-	-	.179	-	-	-	-	.243	-	-
	Chrupała et al. 2017	.055	0.163	.253	-	-	-	-	-	-	-
	Chrupała 2019	-	-	.296	-	-	-	-	-	-	-
	Ours (from scratch)	.018	.063	.101	.288	.428	.024	.072	.124	.332	.458
	Ours (warm-starting g_{aud})	.041	.138	.211	.467	.613	.550	.166	.241	.522	.654
	Ours (warm-starting g_{vis})	.062	.190	.279	.560	.703	.081	.242	.352	.664	.782
	Ours (warm-starting all)	**.139**	**.368**	**.495**	**.781**	**.875**	**.182**	**.435**	**.558**	**.842**	**.910**

Table 2: Retrieval scores on the test set of FACC.

over the baseline, and combining both branches leads to the best overall performance.

In particular, we improve R@10 for caption-to-image from the .296 obtained by Chrupała (2019) by 20% absolute to .495, without using multitask training or pretraining g_{vis} on ImageNet (Deng et al., 2009). The multitask training approach of Chrupała (2019) is complementary to our improvements, so further gains might be obtained by combining these strategies. Furthermore, very deep, residual convolutional neural networks over characters have been shown to perform well for text-based tasks (Conneau et al., 2017). We expect that our strategy of using the same basic architecture across different input types (speech, text and image) can be fruitfully extended to that setting. A related observation: while our results exceed previous results reported on text/image retrieval settings for FACC, we expect that recent advances in text encoding could easily beat those reported numbers.

We also explore very low-data regimes using our pretrained model (see Fig. 3). Using small training subsets randomly drawn from FACC, we report performance as a function of how much data the model sees. With as little as 10% of the original training data (3000 image/spoken caption pairs), the warm-started model performs competitively with a model trained on all training data.

Qualitative evaluation. Once a model is trained, any input (image or spoken caption) can be be used to query the corpus of images and spoken captions for nearest neighbors in the latent space. Figure 4 shows some examples of retrieved nearest neighbors in FACC's test set. Given a spoken caption or

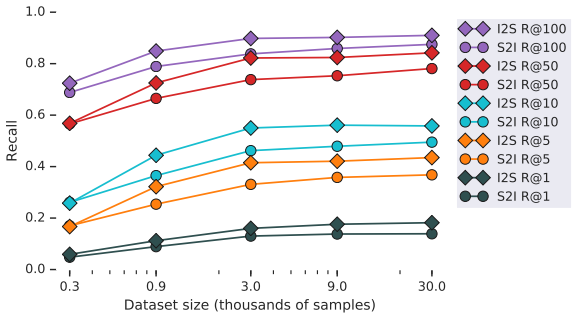


Figure 3: Ablations on low-data regime on FACC: chart shows recall scores for image-to-speech (I2S) and speech-to-image (S2I) retrieval, as a function of the amount of training data used for fine-tuning.

an image we show the five nearest image neighbors and five nearest caption neighbors. From these, it is clear that the representations capture many semantically salient attributes of the inputs. The retrieved items correctly share many thematic elements and many are clearly good matches even though the particular image-caption pairs are not associated in the data. This serves to reinforce our observation that R@k evaluations using only the known paired items is likely to underestimate the actual performance of the models—which we show to be the case with human evaluations in Section 4.4.

Only some items are substantially incompatible: e.g. an image of a car for a caption about a woman in a river (they share water spraying), a picture of three adults for a caption about children raising their hands, and a caption about a boy climbing a wall for an image of children playing leapfrog). That said, many details are poor matches, such as the count of objects (one ball versus many), colors

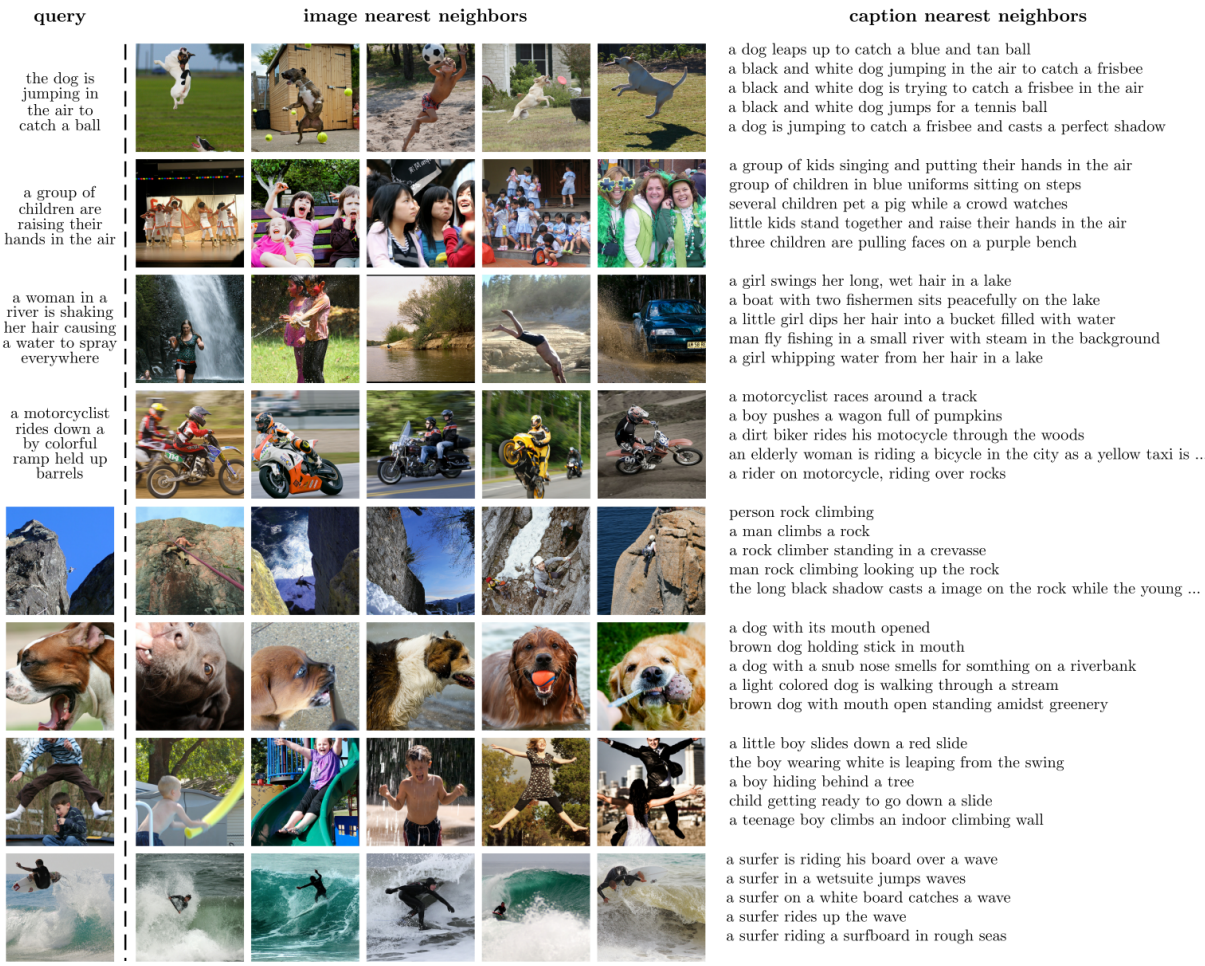


Figure 4: Nearest neighbors in the joint visual and acoustic latent space, best viewed with zoom: using 4 spoken captions and 4 images as queries, we extract from FACC's test set the closest 5 images and 5 spoken captions in the latent space for each of them. For simplicity, we show the text associated with each spoken caption.

(brown dogs versus multicolored ones), people descriptions (elderly woman versus male dirt biker), object identification (e.g. a yellow pool noodle viewed as similar to slides), processes (jumping versus sliding) and perspective (man looking up versus viewed from behind and climbing). As such, there is clearly significant headroom for better, more fine-grained modeling of both captions and images. Additionally, cross-modal attention mechanisms (Xu et al., 2015) and other explainability techniques (Ribeiro et al., 2016) could help better inspect and understand a model's predictions.

Furthermore, as noted by Chrupała et al. (2017), text-based retrieval models often handle misspellings poorly. In contrast, speech-based models are unlikely to suffer from similar problems because they inherently must deal with variation in the expression of words and utterances. For instance, the caption *"a dirt biker rides his **motocycle** through the woods"* (fourth row of Figure 4) is highly correlated with the correctly spelled sentences.

4.4 Human evaluation

We ran human evaluations to answer two questions: (1) how much does cropping limit model performance? and (2) how much do retrieval evaluations based only on positive associations underestimate model performance? Hints about both questions can be seen in the qualitative evaluation (Fig. 4).

To answer the first question, Table 3 shows the ratings for ground truth image/caption pairs in the FACC test set. The *uncropped* row shows that overall the captions are high quality and do match the full images. However, human ratings on images *cropped* at the center (which is what is provided to the models) show that there is considerable loss from cropping—only 62.5% of cropped images are rated as good matches by all five raters. Inspection makes it clear why cropping hurts: for example an

	"good" ratings (out of 5)				
	1+	2+	3+	4+	5
Cropped	.949	.918	.874	.800	.625
Uncropped	.995	.994	.989	.971	.891

Table 3: Human evaluation results on ground truth pairs on the test set of FACC, using either center cropped (which the models receive) or uncropped versions of the images.

image of a snowboarder in the air next to another on a ski lift is cropped such that the snowboarder is missing, and thus a poor match to captions mentioning the snowboarder. This clearly indicates that standard cropping (which we follow) inherently limits model performance and that strategies that use the full image should be explored.

Standard retrieval evaluations are blind to pairs that match but are not associated in the data. To address this and answer the second question posed above, we present the top-5 retrieved captions for each image and the top-5 retrieved images for each caption in FACC's test set to human raters. To increase speed and decrease costs, we show raters the original Flickr8k textual captions instead of the spoken ones. Each pair is judged by five raters as "good" or not. This gives a soft measure of the compatibility of each pair based on fast binary judgments from each rater. For retrieval evaluations of a model, we compute recall based on the majority of human raters approving each image-caption pair: R@1 is the percentage of top-1 results and R@5 the percentage of top-5 results that are evaluated as a match by at least 3 of the 5 raters.

Table 4 shows these metrics computed on retrieval outputs from two settings: FACC training from scratch and FACC fine-tuning after CSC pretraining. It also shows the corresponding automatic evaluations from Table 2 for easy comparison. These results make it clear that evaluation based only on positive associations is too rigid: speech-to-image retrieval based on human evaluations shows that a good matching item is returned in 52.2% of cases rather than just the 36.8% indicated by strict corpus matches. For image-to-speech retrieval the 55.8% strict measure goes up to 63.8%. That said, the results also show that the strict measure is nevertheless a useful indicator for comparing relative model performance: the model pretrained on CSC beats the corresponding one trained on FACC from scratch, on both human and automatic evaluations.

		S2I		I2S	
Eval	Pretrain	R@1	R@5	R@1	R@5
Auto		.018	.063	.024	.072
Auto	✓	.139	.368	.182	.558
Humans		.056	.154	.070	.196
Humans	✓	.229	.522	.306	.638

Table 4: Comparison of human rater scores (majority agreement) versus using only corpus-known pairs on all metrics for speech-to-image (S2I) and image-to-speech (I2S) retrieval. Rows with *Auto* evaluation correspond to *Ours (from scratch)* and *Ours (warm-starting all)* scores in Table 2.

5 Conclusion

Large-scale datasets are essential for training deep networks from scratch. In this paper, we present a scalable method for generating an audio caption dataset taking advantage of TTS systems to create millions of data pairs. Using the MMS loss, we demonstrate that pretraining on this dataset greatly improves performance on a human-generated audio caption dataset. As TTS models continue to improve and be developed for more languages, this data augmentation strategy will only become more robust and helpful over time. Finally, using human evaluations, we show evidence that corpus-based retrieval scores underestimate actual performance.

This present work is focused on the here and now since captions describe a snapshot in time and focus on the visual entities and events involved in them. We thus have little hope to learn representations for words like *visit*, *career* and *justice*, for example. Videos can help with process oriented words like *visit* and could get significant components of words like *career* (such as the visual contexts, but not the overall path with intermediate goals involved in careers). They are likely to be hopeless for abstract words like *justice*. To address problems of this sort, there are likely many opportunities to combine ideas from unsupervised term discovery (Kamper et al., 2016; Bansal et al., 2017) with audiovisual word learning (Harwath et al., 2018) and models of visual grounding that have been applied to text (Kiros et al., 2018). Being able to learn effective representations from raw audio associated with images could provide new possibilities for work that learns from videos and text (transcribed speech) (Chen et al., 2018), and in particular open up such techniques to new languages and domains.

Acknowledgements

The authors would like to thank Radu Soricut, Austin Waters, Alex Ku and Jeffrey Ling for the helpful comments that assisted the development of this work.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, pages 26–33. Association for Computational Linguistics.

Sameer Bansal, Herman Kamper, Sharon Goldwater, and Adam Lopez. 2017. Weakly supervised spoken term discovery using cross-lingual side information. In *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-2017)*.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*, 55:409–442.

Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. 2016. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*, pages 850–865. Springer.

Saurabhch Bhati, Herman Kamper, and K Sri Rama Murty. 2018. Phoneme based embedded segmental k-means for unsupervised term discovery. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5169–5173. IEEE.

Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. 2018. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744.

Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. 2011. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on Psychological Science*, 6(1):3–5.

Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. 2018. Temporally grounding natural sentence in video. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 162–171, Brussels, Belgium. Association for Computational Linguistics.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Grzegorz Chrupała. 2019. Symbolic inductive bias for visually grounded learning of spoken language. In *To appear in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Grzegorz Chrupała, Lieke Gelderloos, and Afra Alishahi. 2017. Representations of language in a model of visually grounded speech signal. *arXiv preprint arXiv:1702.01991*.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1107–1116, Valencia, Spain. Association for Computational Linguistics.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee.

Alon Halevy, Peter Norvig, and Fernando Pereira. 2009. The unreasonable effectiveness of data.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

David Harwath and James Glass. 2015. Deep multimodal semantic embeddings for speech and images. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 237–244. IEEE.

David Harwath and James Glass. 2019. Towards visually grounded sub-word speech unit discovery. *arXiv preprint arXiv:1902.08213*.

David Harwath, Adria Recasens, Dídac Surís, Galen Chuang, Antonio Torralba, and James Glass. 2018. Jointly discovering visual objects and spoken words from raw sensory input. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 649–665.

David Harwath, Antonio Torralba, and James Glass. 2016. Unsupervised learning of spoken language with visual context. In *Advances in Neural Information Processing Systems*, pages 1858–1866.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yunhsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

PW Jusczyk and RN Aslin. 1995. Infants detection of the sound patterns of words in fluent speech. *Cognitive psychology*, 29:1–23.

Herman Kamper, Aren Jansen, and Sharon Goldwater. 2016. Unsupervised word segmentation and lexicon discovery using acoustic word embeddings. *IEEE Transactions on Audio, Speech and Language Processing*, 24(4):669679.

Herman Kamper, Aren Jansen, and Sharon Goldwater. 2017a. A segmental framework for fully-unsupervised large-vocabulary speech recognition. *Computer Speech & Language*, 46:154–174.

Herman Kamper, Shane Settle, Gregory Shakhnarovich, and Karen Livescu. 2017b. Visually grounded learning of keyword prediction from untranscribed speech. *arXiv preprint arXiv:1703.08136*.

Andrej Karpathy, Armand Joulin, and Li F Fei-Fei. 2014. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in neural information processing systems*, pages 1889–1897.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jamie Kiros, William Chan, and Geoffrey Hinton. 2018. Illustrative language understanding: Large-scale visual grounding with image search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 922–933, Melbourne, Australia. Association for Computational Linguistics.

Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M Cohen, Huyen Nguyen, and Ravi Teja Gadde. 2019. Jasper: An end-to-end convolutional neural acoustic model. *arXiv preprint arXiv:1904.03288*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Lucas Ondel, Lukáš Burget, and Jan Černockỳ. 2016. Variational inference for acoustic unit discovery. *Procedia Computer Science*, 81:80–86.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE.

Alex S Park and James R Glass. 2007. Unsupervised pattern discovery in speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(1):186–197.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144. ACM.

Dan Roth. 2017. Incidental supervision: Moving beyond supervised learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Jenny R. Saffran, Elissa L. Newport, and Richard N. Aslin. 1996. Word segmentation: The role of distributional cues. *Journal of Memory and Language*, 35:4:606–621.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. 2018. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852.

Felix Sun, David Harwath, and James Glass. 2016. Look, listen, and decode: Multimodal speech recognition with images. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 573–578. IEEE.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *SSW*, 125.

Balakrishnan Varadarajan, Sanjeev Khudanpur, and Emmanuel Dupoux. 2008. Unsupervised learning of acoustic sub-word units. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 165–168. Association for Computational Linguistics.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Melbourne, Australia. Association for Computational Linguistics.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Yinfei Yang, Gustavo Hernandez Abrego, Steve Yuan, Mandy Guo, Qinlan Shen, Daniel Cer, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. 2019. Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax. *arXiv preprint arXiv:1902.08564*.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations (ICLR)*.

Using Priming to Uncover the Organization of Syntactic Representations in Neural Language Models

Grusha Prasad
Johns Hopkins University
grusha.prasad@jhu.edu

Marten van Schijndel
Cornell University
mv443@cornell.edu

Tal Linzen
Johns Hopkins University
tal.linzen@jhu.edu

Abstract

Neural language models (LMs) perform well on tasks that require sensitivity to syntactic structure. Drawing on the syntactic priming paradigm from psycholinguistics, we propose a novel technique to analyze the representations that enable such success. By establishing a gradient similarity metric between structures, this technique allows us to reconstruct the organization of the LMs' syntactic representational space. We use this technique to demonstrate that LSTM LMs' representations of different types of sentences with relative clauses are organized hierarchically in a linguistically interpretable manner, suggesting that the LMs track abstract properties of the sentence.

1 Introduction

Neural networks trained on text alone, without explicit syntactic supervision, have been surprisingly successful in tasks that require sensitivity to sentence structure. The difficulty of interpreting the learned neural representations that underlie this success has motivated a range of analysis techniques, including diagnostic classifiers (Giulianelli et al., 2018; Conneau et al., 2018; Shi et al., 2016), visualization of individual neuron activations (Kádár et al., 2017; Qian et al., 2016), ablation of individual neurons or sets of neurons (Lakretz et al., 2019) and behavioral tests of generalization to infrequent or held out syntactic structures (Linzen et al., 2016; Weber et al., 2018; McCoy et al., 2018); for reviews, see Belinkov and Glass (2019) and Alishahi et al. (2019).

This paper expands the toolkit of neural network analysis techniques by drawing on the **syntactic priming** paradigm, a central tool in psycholinguistics for analyzing human syntactic representations (Bock, 1986). This paradigm is based on the empirical finding that people tend to reuse syntactic structures that they have recently produced or encountered. For example, English provides two roughly equivalent ways to express a transfer event:

(1) a. The boy threw the ball to the dog.
b. The boy threw the dog the ball.

When readers encounter one of these variants in the text more frequently than the other, they expect that future transfer events will more likely be expressed using the frequent construction than the infrequent one. For example, after reading sentences like (1a) (the **prime**), readers expect sentences like (2a), which shares syntactic structure with the prime, to occur with a greater likelihood than the alternative variant like (2b) which does not (Wells et al., 2009).[1]

(2) a. The lawyer sent the letter to the client.
b. The lawyer sent the client the letter.

We use the priming paradigm to analyze neural network language models (LMs), systems that define a probability distribution over the n^{th} word of a sentence given its first $n-1$ words. Building on paradigms that determine whether the LM's expectations are consistent with the syntactic structure of the sentence (Linzen et al., 2016), we measure the extent to which a LM's expectation for a specific syntactic structure is affected by recent experience with related structures. We prime a fully trained model with a structure by adapting it to a small number of sentences containing that structure (van Schijndel and Linzen, 2018). We then measure the change in surprisal (negative log probability) after adaptation when the LM is tested either on sentences with the same struc-

[1] Wells et al. (2009) measured priming effects for relative clauses, not dative constructions. For work on priming in production with dative constructions, see Kaschak et al. (2011).

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 66–76
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

ture or sentences with different but related structures. The degree to which one structure primes another provides a graded similarity metric between the model's representations of those structures (cf. Branigan and Pickering 2017), which allows us to investigate how the representations of sentences with these structures are organized.

As a case study, we applied this technique to investigate how recurrent neural network (RNN) LMs represent sentences with relative clauses (RCs). We found that the representations of these sentences are organized in a linguistically interpretable manner: sentences with a particular type of RC were most similar to other sentences with the same type of RC in the LMs' representation space. Furthermore, sentences with different types of RCs were more similar to each other than sentences without RCs. We demonstrate that the similarity between sentences was not driven merely by specific words that appeared in the sentence, suggesting that the LMs tracked abstract properties of the sentence. This ability to track abstract properties decreased as the training corpus size increased. Finally, we tested the hypothesis that LMs' accuracy on agreement prediction (Marvin and Linzen, 2018) would increase with the LMs' ability to track more abstract properties of the sentence, but did not find evidence for this hypothesis.

2 Background

2.1 Syntactic predictions in neural LMs

We build on paradigms that use LM probability estimates for words in a given context as a measure of the model's sensitivity to the syntactic structure of the sentence (Linzen et al., 2016; Gulordava et al., 2018; Marvin and Linzen, 2018). If a language model assigns a higher probability to a verb form that agrees in number with the subject (*the boy... writes*) than a verb form that does not (*the boy... write*), we can infer that the model encodes information about the agreement features of nouns and verbs (that is, the difference between singular and plural) and has correctly identified the subject that corresponds to this verb. This reasoning has been extended beyond subject-verb agreement to study whether the predictions of neural LMs are sensitive to a range of other syntactic dependencies, including negative polarity items (Jumelet and Hupkes, 2018), filler-gap dependencies (Wilcox et al., 2018) and reflexive pronoun binding (Futrell et al., 2019).

2.2 Syntactic priming in humans

Syntactic priming has been used to study whether the representations of two sentences have shared structure. For example, (1a) (repeated below as (3)) shares the structure VP → V NP PP with (4a) but not (4b).

(3) The boy threw the ball to the dog.

(4) a. The renowned chef made some wonderful pasta for the guest.
b. The renowned chef made the guest some wonderful pasta.

If (3) primes (4a) more than it primes (4b), we can infer that the representations of (3) are more similar to that of (4a) than to that of (4b). Since (4b) and (4a) differ only in their structure, this difference in similarity must be driven by structural information in the representations of the sentences (for reviews, see Mahowald et al. 2016 and Tooley and Traxler 2010).

Although priming studies have traditionally measured the priming effect on the sentence immediately following the prime, more recent studies have demonstrated that the effects of syntactic priming can be cumulative and long-lasting: sentences with a shared structure S_X become progressively easier to process when preceded by n sentences with the same structure S_X than when preceded by n sentences with a different structure S_Y (Kaschak et al., 2011; Wells et al., 2009).[2] In conjunction with the finding that words that are consistent with a probable syntactic parse are easier to process than words consistent with less probable parses (Hale, 2001; Levy, 2008), the increased ease of processing in cumulative priming studies can be interpreted as evidence that, with increased exposure to a structure, participants begin to expect that structure with a greater probability (Chang et al., 2006).

Cumulative priming allows us to study how sentences are related to each other in the human (or LM) representation space in the same way that non-cumulative priming does: when participants (or LMs) are exposed to sentences with structure S_X, if there is a greater decrease in surprisal when they are tested on other sentences with S_X than when they are tested on other sentences with S_Y, we can infer that the representations of sentences with S_X are more similar to each other than to the

[2] In studies looking at non-cumulative priming, $n = 1$.

Abstract structure	Example
Unreduced Object RC	The conspiracy that the employee welcomed divided the beautiful country.
Reduced Object RC	The conspiracy the employee welcomed divided the beautiful country.
Unreduced Passive RC	The conspiracy that was welcomed by the employee divided the beautiful country.
Reduced Passive RC	The conspiracy welcomed by the employee divided the beautiful country.
Active Subject RC	The employee that welcomed the conspiracy quickly searched the buildings.
PS/ORC-matched Coordination	The conspiracy welcomed the employee and divided the beautiful country.
ASRC-matched Coordination	The employee welcomed the conspiracy and quickly searched the buildings.

Table 1: Examples of sentences generated using templates containing the seven abstract structures we analyzed (optional elements, which only occur in a subset of the examples, are indicated in grey).

representations of sentences with S_Y.

2.3 LM adaptation as cumulative priming

Van Schijndel and Linzen (2018) modeled cumulative priming in recurrent neural networks (RNNs) by adapting fully trained RNN LMs to new stimuli — i.e. taking a fully trained RNN LM and continuing to train it on a small set of sentences (cf. Grave et al. 2017; Krause et al. 2017; Chowdhury and Zamparelli 2019). They demonstrated that when an RNN LM was adapted to a small number of sentences with a shared syntactic structure, the surprisal for novel sentences with that structure decreased, enabling them to infer that the LM's representations of sentences contained information about that structure.

3 Similarity between syntactic structures in RNN LM representational space

Following the assumptions in Section 2.2, we define a similarity metric between two structures S_X and S_Y in an LM's representation space by adapting the LM to sentences with S_X and measuring the change in surprisal for sentences with S_Y — i.e. measuring to what extent sentences with S_X prime sentences with S_Y. We use the notation $\mathbb{A}(Y \mid X)$ to refer to this change in surprisal[3], where X and Y are non-lexically-overlapping sets of sentences whose members share the structures S_X and S_Y respectively. If we assume that S_X and S_Y are similar to each other in the LM's representation space, then $\mathbb{A}(Y \mid X) > 0$ — i.e., encountering sentences with S_X causes the LM to assign a higher probability to sentences with S_Y. On the other hand, if we assume that S_X and S_Y are unrelated to each other, then $\mathbb{A}(Y \mid X) = 0$ — i.e., encountering sentences with S_X does not cause the LM to change its probability for sentences with S_Y.

[3] $\mathbb{A}$ is shorthand for adaptation.

4 Experimental setup

4.1 Syntactic structures

We analyzed five types of RCs. In an active subject RC, the gap is in the subject position of the embedded clause:[4]

(5) My cousin that _ liked the book ...

In a passive subject RC (*passive RCs*), the gap is in the subject position of the embedded clause, and the embedded verb is passive. In English, passive RCs can be unreduced (6a) or reduced (6b):

(6) a. The book that _ was liked by my cousin ...
b. The book _ liked by my cousin ...

In an object RC the gap is in the object position of the embedded clause. In English, object RCs can be unreduced (7a) or reduced (7b):

(7) a. The book that my cousin liked _ ...
b. The book my cousin liked _ ...

Finally, we also included two additional conditions with verb coordination: one with nearly identical word order and lexical content as active subject RCs ((8); ASRC-matched Coordination), and another with nearly identical word order and lexical content as passive RCs and object RCs ((9); PS/ORC-matched Coordination).[5]

(8) My cousin liked the book and ...

(9) The book liked my cousin and ...

[4] We illustrate the location of the gap with underscores here, but the underscores were not included in the LM's input.

[5] In order to maintain the same word order as in object and passive RCs, the subject of the coordinated verb phrases is an NP that tends to fill the object position in other sentences (e.g, "the equation"). Therefore, many of the sentences in this condition are implausible (e.g., "The equation reviewed the physicists and challenged the method.")

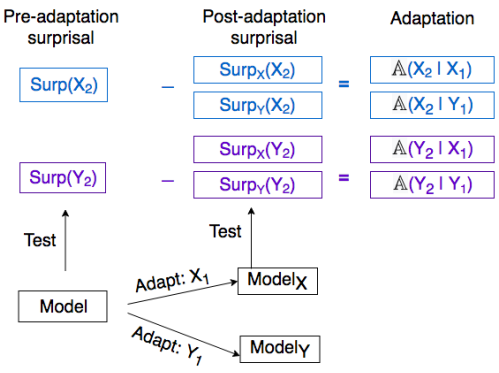


Figure 1: A schematic for calculating the similarity between two structures S_X and S_Y in an LM's representation space. X_1, X_2 and Y_1, Y_2 are non-lexically-overlapping sets of sentences with S_X and S_Y respectively. $Model_X$ and $Model_Y$ refer to versions of a fully trained model that have been adapted to either X_1 or Y_1 respectively. $Surp_X()$ and $Surp_Y()$ are functions that return the surprisal of sentences for $Model_X$ and $Model_Y$.

These conditions enable us to measure whether sentences with different types of RCs are more similar to each other in an LM's representation space than they are to lexically matched sentences without RCs.

4.2 Adaptation and test sets

We generated sentences from seven templates, one for each of the syntactic structures of interest. The slots were filled with 223 verbs, 164 nouns, 24 adverbs and 78 adjectives such that the semantic plausibility of the combination of nouns, verbs, adverbs and adjectives was ensured. The seven variants of every sentence had nearly identical lexical items (see Table 1).[6] We used these templates to generate five experimental lists — each list comprised of a pair of adaptation and test sets with minimal lexical overlap between them (only function words and some modifiers were shared). Each adaptation set contained 20 sentences and each test set contained 50.

In order to infer that any decrease in surprisal is caused by adaptation to an abstract syntactic structure, we need to ensure that the models are not adapting to properties of the sentence that are unrelated to the abstract structure of interest. Consider a LM adapted to (10) and tested on (11):

(10) The conspiracy that the employee welcomed divided the country.

(11) The proposal that the receptionist managed shocked the CEO.

When the LM is adapted to sentences such as (10), it could adjust its expectations about several properties of the sentence, some more linguistically interesting than others. For instance, it could learn that there are three determiners in the sentence, that the third word of the sentence is *that*, that sentences have nine words, that every verb is preceded by a noun, and so on and so forth. If there is a decrease in surprisal when a model is adapted to (10) and tested on (11), it is unclear if this is because the model learned to expect object relative clauses or if it learned to expect any of the other mentioned properties.

To minimize the likelihood that the adaptation effects are driven by irrelevant properties of the sentence, we introduced several sources of variability to our templates: nouns could either be singular or plural, noun phrases could be optionally modified by an adjective, adjectives were optionally modified with an intensifier and verb phrases were optionally modified with adverbs which could occur either pre-verbally or post-verbally (details in the Supplementary Materials).[7]

4.3 Models

We used 75 of the LSTM language models trained by van Schijndel et al. (2019); these LMs varied in the number of hidden units per layer (100, 200, 400, 800, 1600) and the number of tokens they were trained on (2 million, 10 million or 20 million). For each training corpus size, van Schijndel and Linzen trained models on five disjoint subsets of the WikiText-103 corpus, to ensure that the results generalized across different training sets.

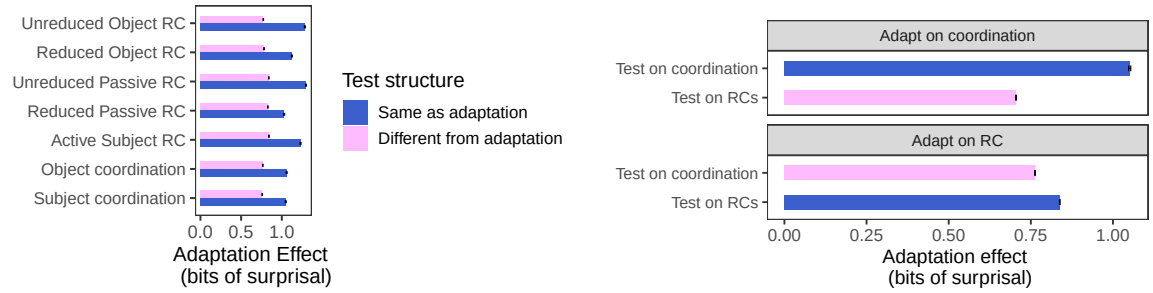
4.4 Calculating the adaptation effect (AE)

For every structure, we computed the similarity between that structure and every other structure (including itself) as described in Section 3. This process is schematized in Figure 1. The surprisal values were averaged across the entire sentence.[8]

[6] Since the main verb of the sentence was constrained to be semantically plausible with the subject of the sentence, it often varied between active subject RC and ASRC-matched coordination on the one had and all other conditions on the other.

[7] The Supplementary Materials, the templates and and code for all the analyses along with the data can be found on GitHub: https://github.com/grushaprasad/RNN-Priming

[8] Unknown words were excluded from this average.



(a) (b)

Figure 2: The adaptation effect averaged across all 75 models when (a) they were adapted to each of the structures and tested on either the same structure (blue, bottom) or different structure (pink, top) and (b) they were adapted to RCs and tested on non-RCs or vice versa (pink bars); or when they were adapted to RCs or non-RCs and tested on other RCs or and non-RCs respectively (blue bars). Greater values indicate more similarity between adaptation and test structures. Error bars reflect 95% CIs.

We found that $\mathbb{A}(B \mid A)$ was proportional to the surprisal of B prior to adaptation (see Supplementary Materials). As a consequence, for three structures X, Y and Z, $\mathbb{A}(Y \mid X)$ could be greater than $\mathbb{A}(Z \mid X)$ merely because Y was a more surprising structure to begin with than Z. In order to remove this confound, we first fit a linear regression model predicting $\mathbb{A}(Y \mid X)$ from the surprisal of Y prior to adaptation ($\mathit{Surp}(Y)$):

$$\mathbb{A}(Y \mid X) = \beta_0 + \beta_1 \mathit{Surp}(Y) + \epsilon$$

We then regressed out the linear relationship between $\mathbb{A}(Y \mid X)$ and $\mathit{Surp}(Y)$ as follows:

$$\begin{aligned} \mathit{AE}(Y \mid X) &= \mathbb{A}(Y \mid X) - \beta_1 \mathit{Surp}(Y) \\ &= \beta_0 + \epsilon \end{aligned}$$

Since $\mathit{Surp}(Y)$ was centered around its mean, β_0 reflects the mean of $\mathbb{A}(Y \mid X)$ when $\mathit{Surp}(Y)$ is equal to the mean surprisal of all sentences prior to adaptation. The term ϵ reflects any variance in $\mathbb{A}(Y \mid X)$ that is not predicted by $\mathit{Surp}(Y)$. By summing these two terms together, $\mathit{AE}(Y \mid X)$ reflects the change in surprisal for Y after adapting to X that is independent of $\mathit{Surp}(Y)$.

4.5 Statistical analyses

We used linear mixed effects models (Pinheiro et al., 2000) to test for statistical significance; all of the results reported below were highly significant. Details about the statistical analyses can be found in the Supplementary Materials.

5 Results

5.1 Validating AE as a similarity metric

As discussed in Section 2.3, under the adaptation-as-priming paradigm, we would expect sentences that share the same specific structure to be more similar to each other than lexically matched sentences that do not share the structure.[9] In other words, if X_1 and X_2 are non-lexically-overlapping sets of sentences with shared structure S_X, and Y_2 is a set of sentences with structure S_Y, but is lexically matched with X_2, then we would expect $AE(X_2 \mid X_1) > AE(Y_2 \mid X_1)$. We found this prediction to be true for all of our seven structures (Figure 2a), thus validating our similarity metric.

5.2 Similarity between sentences with different types of VP coordination

Our two coordination conditions were structurally identical to each other but varied in their semantic plausibility — the sentences in PS/ORC-matched coordination condition were often semantically implausible whereas sentences in ASRC-matched condition were always semantically plausible (see footnote 5). If sentences that were structurally similar were close together irrespective of semantic plausibility, then we expect sentences with coordination to be more similar to each other than lexically matched sentences with RCs. Consistent with this prediction, the adaptation effect for models adapted to one type of coordination was greater when the models were tested on sentences with the other type of coordination than when they were tested on sentences with RCs (top panel of Figure 2b).

[9] By lexically matched we mean that all content words were shared between sentences.

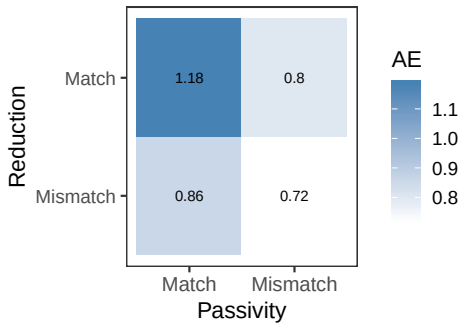


Figure 3: The adaptation effect when models adapted to sentences with reduced and unreduced RCs are tested on sentences that match only in reduction (top right), match only in passivity (bottom right), match in both reduction and passivity (top left) or sentences that match in neither (bottom right).

5.3 Similarity between sentences with different types of RCs

Unlike sentences with coordination, sentences with different types of RCs differ from each other at a surface level (see Table 1). However, at a more abstract level they all share a common property: a gap. If the RNN LMs were keeping track of whether or not a sentence contained a gap, we would expect sentences with different types of RCs to be more similar to each other in the RNN LMs' representation space than lexically matched sentences without a gap. In other words, if RC_X and RC_Y are two different types of RCs and $Coord_Y$ is a sentence with verb coordination lexically matched with RC_Y, then we would expect $AE(RC_Y \mid RC_X) > AE(Coord_Y \mid RC_X)$.

Consistent with this prediction, the adaptation effect for models adapted to RCs was greater when they were tested on sentences with other types of RCs than when they were tested on sentences with coordination (bottom panel of Figure 2b). This suggests that the LMs do keep track of whether or not a sentence contains a gap, even though this property is not overtly indicated by a lexical item that is shared across all types of RCs.

5.4 Similarity between sentences belonging to different sub-classes of RCs

The different types of RCs we tested can be divided into sub-classes based on at least two linguistically interpretable features: reduction and passivity. Reduction distinguishes reduced passive and object RCs on the one hand from unreduced passive and object RCs on the other. Passivity distinguishes reduced and unreduced passive RCs on the one hand from reduced and unreduced object RCs on the other. The LMs could be tracking either, both or none of these features.

We probed whether the LMs track these features by comparing the similarity between sentences that share one feature but not the other, with the similarity between sentences that share neither feature. If the adaptation effect is greater when there is a match in one feature than when there is a match in neither of the features, we can infer that the LMs track whether sentences have that feature. We found that the LMs track both of these features (Figure 3).

Additionally, we probed which of the features contributes more towards the similarity between sentences by comparing the similarity between sentences that match only in passivity with sentences that match only in reduction. When the adaptation and test sets matched only in passivity, the adaptation effect was slightly (but significantly) greater than when the adaptation and test sets matched only in reduction (Figure 3). In other words, in the LMs' representation space, (12) is more similar to (13) than it is to (14), suggesting that passivity contributes more towards the similarity between sentences than reduction.

(12) The conspiracy the employee welcomed divided the country.

(13) The conspiracy that the employee welcomed divided the country.

(14) The conspiracy welcomed by the employee divided the country.

This result is both intuitive and linguistically interpretable — the edit distance between reduced and unreduced RCs is smaller than the that between object and passive RCs; the syntax tree for (12) is also more similar to (13) than it is to (14).

5.5 What properties of sentences drive the similarity between them?

Our analyses so far have demonstrated that sentences that belong to linguistically interpretable classes (e.g., sentences that match in reduction) are more similar to each other in the LMs' representation space than they are to sentences that do not belong to those classes (e.g., sentences that do not match in reduction). However, it is unclear what properties of the sentences are driving this similarity between members of the class. For al-

most all of the linguistically interpretable classes we considered, all sentences belonging to a class shared at least some, if not all, function words. The only exception was the class of all RCs, where the property shared by all sentences in this class (the presence of a gap) was not overtly observable. Therefore, it is possible that the similarity between members of most of the classes we tested was being driven entirely by the presence of these function words.

In order to test whether the similarity between members of classes was indeed being driven by the presence of shared function words, we compared the representation space of the models we tested in the previous sections (henceforth *trained models*) with the representation space of models trained on no data (henceforth *baseline models*). Since the baseline models were only ever exposed to the 20 sentences in the adaptation set and there was no lexical overlap in content words between adaptation and test sets, any similarity between sentences in the representation space of these models would be driven by the presence of function words. If the similarity between sentences in the representation space of the trained models was being driven by factors other than the presence of function words, we would expect this similarity to be greater than the similarity between these sentences in the representation space of the baseline models.

We cannot directly use adaptation effect to compare the similarity between sentences in the representation spaces of trained models and baseline models, however: models trained on more data are likely to have stronger priors and are therefore less likely to drastically change their representations after 20 sentences than models trained on less data. In order to mitigate this issue, we defined a distance measure between sentences that belong to a class and sentences that do not belong to a class S_X as follows (see Figure 4 for a schematic):

$$\mathbb{D}(S_X, \neg S_X) = \frac{AE(X_2 \mid X_1)}{AE(\neg X_2 \mid X_1)}$$

This value would be greater than one if sentences that belonged to a class were more similar to each other than they were to sentences that did not belong to the class. Since the strength of prior belief would affect sentences that belong to the class the same way it would affect sentences that do not belong to the class, the effect would cancel out.

We measured the distance between members and non-members for three linguistically inter-

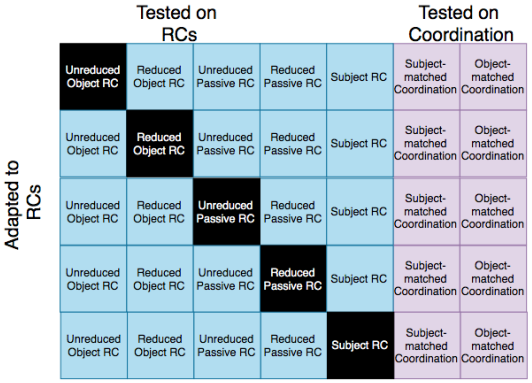
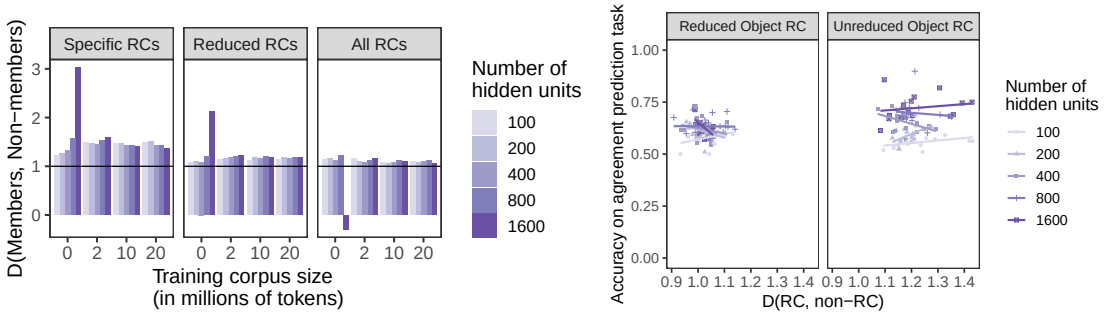


Figure 4: A schematic of how $\mathbb{D}(RC, \neg RC)$ is calculated. For any given row, the black square indicates the specific structure the models were adapted to, the blue squares indicate other structures that belong to the same linguistically defined class as the black square and the pink squares indicate the structures that do not belong to this linguistically defined class. In calculating the distance, we first calculated the proportion between the mean adaptation effect for the blue squares and the mean adaptation effect for pink squares for each row. We then averaged across the proportion for each row to arrive at one number.

pretable classes: sentences which contained the same type of RC, sentences that matched in their reduction or sentences that contained any type of RC. In our baseline models, for all three classes, sentences that belonged to one of these classes were more similar to each other than sentences that did not belong to that class (Figure 5a). This was surprising for the class of sentences that contained any type of RC because there was no function word that was shared by all sentences in this class. We hypothesize that this is because sentences without RCs always contained the word *and*, whereas sentences with RCs never did.

In cases where members of the class shared at least some function words, the distance between sentences that belonged to the class and sentences that did not for the trained models was greater than that for the baseline models. This suggests that the similarity between sentences in the representation space of trained models was being driven by factors other than the mere presence of function words. However, somewhat surprisingly, as the number of training tokens increased, the distance between members and non-members decreased.

In the case where the members of the class did not share any function words, the distance between sentences that belonged to the class and sentences that did not belong to the class did not differ be-



(a) (b)

Figure 5: (a) Effect of hidden layer size and corpus size on the distance between sentences with specific RCs and sentences without (left), between sentences that match in reduction and sentences that do not (middle) and between sentences with RCs and sentences without (right). The solid black line indicates the point at which sentences that belong to a particular class are equally similar to other sentences that belong to that class and sentences that do not. (b) Agreement prediction accuracy on reduced object RCs and unreduced object RCs as a function of $\mathbb{D}(RC, \neg RC)$

tween the trained models and the baseline models. This suggests that any similarity between sentences in the representation space of trained models was driven purely by the presence (or in this case absence) of lexical items.

5.6 Does $\mathbb{D}(RC, \neg RC)$ predict agreement prediction accuracy?

Marvin and Linzen (2018) created a dataset that evaluated the grammaticality of the predictions of language models. Using this dataset, they showed that LSTM LMs could not accurately predict the number of the main verb if the main clause subject was modified by an object RCs (either reduced or unreduced). However, the models had better performance if the main clause was modified by an active subject RC. For example, the models were at near chance levels in predicting that (15a) should have higher probability than (15b), but were slightly better at predicting that (16a) should have higher probability than (16b):

(15) a. The farmer that the parents love swims.
b. *The farmer that the parents love swim.

(16) a. The farmer that loves the parents swims.
b. *The farmer that loves the parents swim.

One possible explanation for this poor performance is that object RCs, either reduced or unreduced, are quite infrequent (Roland et al., 2007). If the LM treats object RCs as unrelated to other RCs, there are likely very few training examples from which the models can learn about subject-verb agreement when the subject is modified by an object RC. If the LM had instead treated object RCs as belonging to the same class as other RCs, it could learn to generalize from training examples of subject-verb agreement when the subject is modified by other RCs. This suggests the hypothesis that agreement prediction accuracy on object RCs will be higher in LMs in which the representation of object RCs is more similar to the representation of other RCs.

The similarity between object RCs and other RCs was defined as in the previous section (the proportion of blue squares to pink squares of the top two rows in Figure 4). There was an increase in accuracy as the number of hidden units increased (see Figure 5b). However, the similarity between object RCs and other types of RCs did not significantly correlate with agreement prediction; we therefore did not find any evidence for the hypothesis mentioned above.[10]

6 Discussion

Drawing on the syntactic priming paradigm from psycholinguistics, we proposed a new technique to analyze how the representations of sentences in neural language models (LMs) are organized. Applying this paradigm to sentences with relative clauses (RCs), we found that the representations of these sentences were organized in a linguistically interpretable hierarchical manner (summarized in Figure 6).

We investigated whether this hierarchical organization was driven by function words that are shared among sentences sentences or whether there was evidence that LMs were tracking more

[10] Similar patterns were observed for the other constructions in the dataset. See Supplementary Materials.

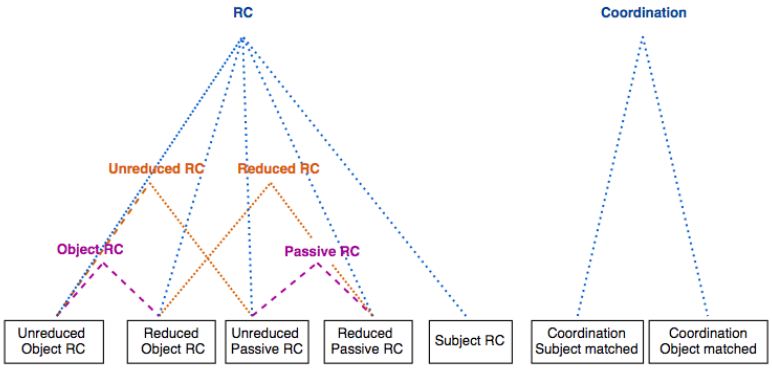


Figure 6: A schematic of how sentences belonging to different linguistically defined classes are related to each other in the LMs' representation space. Each colour indicates a different level of hierarchy.

abstract properties of the sentence. We found that for at least some linguistically interpretable classes, sentences that belonged to these classes were more similar to each other in the representation space of the LMs we tested than in the representation space of baseline LMs that were not trained on any data. This suggests that the trained LMs were capable of tracking abstract properties of the sentence.

However, for linguistically interpretable classes in which sentences shared a non-lexically observable property (e.g. presence of a gap), sentences were as similar to each other in the representation space of the LMs we tested as in the representation space of baseline LMs. Taken together, these results suggest that LMs might be able to track abstract properties of classes of sentences only if these classes also share a lexically observable property.

Additionally, we found that the sentences belonging to linguistically interpretable classes were more similar to each other in the representation spaces of models trained on 2 million tokens than in the representation spaces for models trained on 20 million tokens. We infer from this that LMs' ability to track abstract properties of sentences decreases with an increase in the training corpus size. This suggests that if we want these LMs to track more abstract linguistic properties, training them on more data from the same distribution is unlikely to help (cf. van Schijndel et al. 2019). Future work can explore how to bias these models to track linguistically useful properties through architectural biases (Dyer et al., 2016), training on auxiliary tasks (Enguehard et al., 2017) or data augmentation (Perez and Wang, 2017).

We hypothesized that models' accuracy on subject verb agreement when preceded by object RCs would increase as the similarity between object RCs and the other types of RCs increased. However, we did not find evidence for this. This could either be because the similarity between object RCs and the other types of RCs was too weak to be useful (see Figure 5a) or because the LMs do not use this property when predicting verb agreement. Future work can disambiguate these reasons by testing models that are biased to treat sentences with object RCs and other RCs as being similar.

Finally, our method allows us to generate a similarity matrix in the LMs representation space for any given set of structures. In the future, generating a similar matrix for human representations using priming experiments and comparing these two matrices using analysis methods from cognitive neuroscience (Kriegeskorte et al., 2008) may enable us to gain insight into how human-like the LM representations are and vice versa.

7 Conclusion

We proposed a novel technique to analyze how the representations of various syntactic structures are organized in neural language models. As a case study, we applied this technique to gain insight into the representations of sentences with relative clauses in RNN language models and found that the representations of sentences were organized in a linguistically interpretable manner.

8 Acknowledgments

We would like to thank Sadhwi Srinivas and the members of the CAP lab at JHU for helpful discussions and valuable feedback.

References

Afra Alishahi, Grzegorz Chrupała, and Tal Linzen. 2019. Analyzing and interpreting neural networks for NLP: A report on the first BlackboxNLP workshop. *Journal of Natural Language Engineering*, 25(4):543–557.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

J. Kathryn Bock. 1986. Syntactic persistence in language production. *Cognitive Psychology*, 18(3):355–387.

Holly P. Branigan and Martin J. Pickering. 2017. An experimental approach to linguistic representation. *Behavioral and Brain Sciences*, 40.

Franklin Chang, Gary S. Dell, and Kathryn Bock. 2006. Becoming syntactic. *Psychological Review*, 113(2):234.

Shammur Absar Chowdhury and Roberto Zamparelli. 2019. An LSTM adaptation study of (un)grammaticality. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 204–212, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Émile Enguehard, Yoav Goldberg, and Tal Linzen. 2017. Exploring the syntactic abilities of RNNs with multi-task learning. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 3–14, Vancouver, Canada. Association for Computational Linguistics.

Richard Futrell, Ethan Wilcox, Takashi Morita, Peng Qian, Miguel Ballesteros, and Roger Levy. 2019. Neural language models as psycholinguistic subjects: Representations of syntactic state. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 32–42, Minneapolis, Minnesota. Association for Computational Linguistics.

Mario Giulianelli, Jack Harding, Florian Mohnert, Dieuwke Hupkes, and Willem Zuidema. 2018. Under the hood: Using diagnostic classifiers to investigate and improve how language models track agreement information. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 240–248, Brussels, Belgium. Association for Computational Linguistics.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. In Yoshua Bengio and Yann LeCun, editors, *Proceedings of the Fifth International Conference on Learning Representations*. International Conference on Learning Representations.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, pages 1–8, Pittsburgh, PA. Association for Computational Linguistics.

Jaap Jumelet and Dieuwke Hupkes. 2018. Do language models understand anything? on the ability of LSTMs to understand negative polarity items. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 222–231, Brussels, Belgium. Association for Computational Linguistics.

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780.

Michael P. Kaschak, Timothy J. Kutta, and John L. Jones. 2011. Structural priming as implicit learning: Cumulative priming effects and individual differences. *Psychonomic Bulletin & Review*, 18(6):1133–1139.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2017. Dynamic evaluation of neural sequence models. Technical report, University of Edinburgh.

Nikolaus Kriegeskorte, Marieke Mur, and Peter A. Bandettini. 2008. Representational similarity

analysis-connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2:4.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 11–20, Minneapolis, Minnesota. Association for Computational Linguistics.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106:1126–1177.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Kyle Mahowald, Ariel James, Richard Futrell, and Edward Gibson. 2016. A meta-analysis of syntactic priming in language production. *Journal of Memory and Language*, 91:5–27.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

R. Thomas McCoy, Robert Frank, and Tal Linzen. 2018. Revisiting the poverty of the stimulus: Hierarchical generalization without a hierarchical bias in recurrent neural networks. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 2093—2098, Austin, TX.

Luis Perez and Jason Wang. 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

José Pinheiro, Douglas Bates, et al. 2000. *Mixed-Effects Models in S and S-PLUS*. Springer Science & Business Media.

Peng Qian, Xipeng Qiu, and Xuanjing Huang. 2016. Analyzing linguistic knowledge in sequential model of sentence. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 826–835, Austin, Texas. Association for Computational Linguistics.

Douglas Roland, Fredric Dick, and Jeffrey L. Elman. 2007. Frequency of basic english grammatical structures: A corpus analysis. *Journal of Memory and Language*, 57(3):348–379.

Marten van Schijndel and Tal Linzen. 2018. A neural model of adaptation in reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4704–4710, Brussels, Belgium. Association for Computational Linguistics.

Marten van Schijndel, Aaron Mueller, and Tal Linzen. 2019. Quantity doesnt buy quality syntax with neural language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Hong Kong, China. Association for Computational Linguistics.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Kristen M Tooley and Matthew J Traxler. 2010. Syntactic priming effects in comprehension: A critical review. *Language and Linguistics Compass*, 4(10):925–937.

Noah Weber, Leena Shekhar, and Niranjan Balasubramanian. 2018. The fine line between linguistic generalization and failure in Seq2Seq-attention models. In *Proceedings of the Workshop on Generalization in the Age of Deep Learning*, pages 24–27, New Orleans, Louisiana. Association for Computational Linguistics.

Justine B. Wells, Morten H. Christiansen, David S. Race, Daniel J. Acheson, and Maryellen C. MacDonald. 2009. Experience and sentence processing: Statistical learning and relative clause comprehension. *Cognitive Psychology*, 58:250–271.

Ethan Wilcox, Roger Levy, Takashi Morita, and Richard Futrell. 2018. What do RNN language models learn about filler–gap dependencies? In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 211–221, Brussels, Belgium. Association for Computational Linguistics.

Say *Anything*: Automatic Semantic Infelicity Detection in L2 English Indefinite Pronouns

Ella Rabinovich[1] **Julia Watson**[1] **Barend Beekhuizen**[2] **Suzanne Stevenson**[1]
[1]Dept. of Computer Science, University of Toronto, Canada
[2]Dept. of Language Studies, University of Toronto, Canada
{ella,jwatson,suzanne}@cs.toronto.edu
barendbeekhuizen@utoronto.ca

Abstract

Computational research on error detection in second language speakers has mainly addressed clear grammatical anomalies typical to learners at the beginner-to-intermediate level. We focus instead on acquisition of subtle semantic nuances of English indefinite pronouns by non-native speakers at varying levels of proficiency. We first lay out theoretical, linguistically motivated hypotheses, and supporting empirical evidence on the nature of the challenges posed by indefinite pronouns to English learners. We then suggest and evaluate an automatic approach for detection of atypical usage patterns, demonstrating that deep learning architectures are promising for this task involving nuanced semantic anomalies.

1 Introduction

The ubiquity of English as an online lingua franca offers a rich opportunity for computational research on second language acquisition and on tools for aiding non-native speakers. Most computational research in second language (L2) has focused on spelling and grammar errors, and has been conducted on learners with beginner-to-intermediate proficiency level (henceforth, "learners") (e.g. Ji et al., 2017; Sakaguchi et al., 2017; Rozovskaya et al., 2017; Lo et al., 2018). Little empirical work has looked at *semantic* errors, with existing research mostly focusing on collocations (e.g., Dahlmeier and Ng, 2011; Vecchi et al., 2011; Kochmar and Briscoe, 2013). Also, highly proficient, advanced L2 speakers (henceforth, "advanced L2s") have received little attention (though see Daudaravicius et al., 2016). In contrast to learners, these speakers rarely violate grammatical norms of the L2, but rather deviate from native usage in much more nuanced ways, often exhibiting mild infelicities rather than outright errors.

We aim to explore an elusive aspect of mastering the subtle contours of a word's meaning that are shaped by its context. Specifically, we investigate patterns of acquisition of English indefinite pronouns by L2 speakers. *Indefinite pronouns* (IPs) are linguistic devices that refer to an entity (such as a person or thing) that has not yet been introduced in discourse. In English, examples are words like *someone*, *anything*, and *nobody*. Consider the following sentences, taken verbatim from corpora of L2 speakers (original pronoun is boldfaced; less felicitous usages marked with '?').[1]

1. *Do you know* ***someone****/anyone who was discriminated based on gender?*
2. *It was a little amazing, because they didn't stole* ***?something****/anything.*
3. ***??Anyone****/Someone told me the company has millions in debts and isn't able to pay it.*

Clearly, mastery of IPs in English relies on recognizing subtle factors that determine their appropriate usage in various contexts.

Here, in Section 2, we develop a linguistic analysis with detailed hypotheses on precisely how the tangled relations between *some-* and *any-* pronouns, exemplified above, pose a challenge for L2 learners. In Sections 3 and 4, we perform a large-scale investigation of these linguistic predictions using productions of both learners and advanced L2s, and find that the predicted infelicities occur not only in the language of the former but also the latter, albeit (as expected) to a lesser extent.

A practical goal of this work is to gain predictive power regarding the nuanced semantic difficulties that L2 speakers face. As a first step in that direction, in Section 5 we consider the ability of deep learning language models (LMs) – shown to be adept at capturing grammatical phenomena (Ji

[1]We refer to either less preferred or unacceptable occurrences of an IP, as in (2) and (3), as infelicitous usages.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 77–86
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

Usage class	*some*-?	*any*-?	Example
specific (SP)	✓		I had to reevaluate things when **someone** pointed that out.
non-specific (NS)	✓		**Someone** please make me a GIF of that Wade dunk.
question (QU)	✓	✓	**Anyone** know what the issue might be?
conditional (CD)	✓	✓	I would love it if **someone** could explain it in a more precise way.
indirect negation (IN)	✓	✓	I don't understand how **anyone** can really hate on him.
direct negation (DN)	✓	✓	I don't have **anything** to add other than to say thanks for typing this out.
comparison (CP)	✓	✓	If you work harder you deserve to earn more than **someone** who doesn't do so.
free choice (FC)		✓	...they invite **anyone** on, including musicians sometimes.

Table 1: Usage classes of IPs, an indication of those subsumed by *some*- and *any*-, and examples from our corpora.

et al., 2017; Sakaguchi et al., 2017; Marvin and Linzen, 2018; Goldberg, 2019) – to identify the subtle infelicities that stem from the semantic confusion introduced by *some*- and *any*- IPs. We show that while state-of-the-art models obtain encouraging initial results on this task, they leave room for future improvement (possibly informed by our linguistic findings) in mastering the semantic nuances of the system of English IPs.

The contribution of this work is thus three-fold: First, to our knowledge, we develop the first large-scale empirical investigation of second-language acquisition of indefinite pronouns, constituting a case study of taking a computational approach in linguistic analysis to yield novel insights into challenges in L2 acquisition. Second, we suggest and evaluate an automatic approach to detect infelicities stemming from these challenges in a large collection of L2 productions. Finally, in both cases, we extend our experiments to utterances of highly proficient L2 speakers – a population that has heretofore received little attention in the context of automatic error/infelicity detection.[2]

2 Linguistic Insights into English IPs

Previous work has suggested that the English system of IPs is crosslinguistically atypical, with precise analogues to *some*- and *any*- unusual across languages (Haspelmath, 1997; Beekhuizen et al., 2017). Building on a suggestion from Beekhuizen et al. (2017), we analyze the factors that could lead to difficulty in learning these IPs, and develop detailed hypotheses concerning the challenges that L2 speakers are predicted to face.

Our analysis is based on patterns of *colexification* (Franois, 2008): that is, how usages expressing different semantics are grouped (or not) in various combinations under a single word. As the basis for our analysis, we first need to specify the allowable semantic and syntactic usages of IPs. These *usage classes* are adapted from Haspelmath (1997), who outlines a universal set of IP semantic functions across all languages.[3] Our usage classes are shown in Table 1, with an indication of the classes that *some*- and *any*- can express.

Table 1 illustrates a striking fact about colexification of the usage classes in English: *some*- and *any*- each cover a very broad range of classes, with a high degree of overlap. This level of overlap in languages appears to be very rare: in the 40 languages studied by Haspelmath (1997), we find that only some 10% of languages have IPs that overlap over such a broad area of the semantic space.[4]

Within any of these classes, some semantic/syntactic contexts call for just one of *some*- or *any*-, while others allow both, but with differing meanings (and frequencies/preferences). For example, these similar contexts allow both, but the preferred pronoun differs:

1. *...people care a lot* ***if something is a repost****...*
2. *...before you know* ***if anything is wrong****...*

We thus predict a difficulty for English L2 speakers in having to choose between two (not interchangeable) terms that can be used in highly similar semantic/syntactic environments.

In addition to looking at difficulties posed by the colexification of IPs *within English*, we can consider *crosslinguistic* patterns of colexification for further insight. Semantic typologists have proposed (and empirically supported, across many domains) that the more two underlying concepts are colexified across languages, the more similar those two concepts are (e.g., Anderson, 1982). In

[2] All code and data are available at https://github.com/ellarabi/indefinite-pronouns

[3] Haspelmath's functions are determined by syntactic, semantic, and pragmatic factors. Our usage classes emphasize the syntactic context, for ease of automatic identification and consistent annotation.

[4] Computed using the original Haspelmath's mapping into functions, therefore, not strictly comparable to the slightly different notation of usage classes in this work.

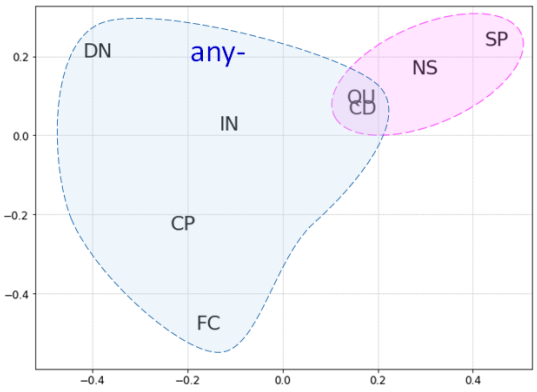


Figure 1: Layout of usage classes in crosslinguistic semantic space; light blue illustrates the scope of English *any-*, pink illustrates the natural grouping of QU/CD with SP/NS.

this way, crosslinguistic patterns of colexification can be used to deduce pairwise similarity among concepts, yielding a universal semantic similarity space for a domain (e.g., Berlin and Kay, 1969; Levinson et al., 2003).

Here, we derive such a similarity space over the IP usage classes of Table 1, using the colexification data across 40 languages, from Haspelmath (1997).[5] We form a distance matrix (found in supplemental materials, A.1) by recording, for every pair of usage classes, the number of languages that have a term subsuming both those classes (indicating their relative similarity). We then use Multidimensional Scaling (MDS) to project the space onto two dimensions, as exemplified in Figure 1.[6]

Figure 1 demonstrates, first, that SP, FC, and DN form three natural "extremes" of the semantic space. In English, these correspond to the canonical uses of the IPs *some-*, *any-*, and *no-*, respectively; thus *some-* is anchored at SP and *any-* at FC (cf. Table 1). Moreover, we find that the usage classes of QU and CD are very close to SP and NS, indicating that QU and CD are most frequently colexified with SP/NS, in particular, much more so than with FC. For English, this means that it is much more natural for *some-* to express QU/CD than for *any-* to do so.

To summarize, our linguistic analysis reveals two potential challenges of English *some-* and *any-*: their confusability across many classes, and the particular difficulty of *any-* in the QU/CD classes. We further find empirically that *some-* IPs are more frequent than *any-* in native English text, suggesting that *some-* will be easier for L2 speakers, and that they may overgeneralize it when faced with uncertainty of which pronoun to use. Collectively, these findings motivate:

Hypothesis 1: The unusually large and overlapping extents of *some-* and *any-* are expected to pose difficulty for L2 speakers; *any-* is predicted to be especially difficult due to its lower frequency.

Hypothesis 2: Due to greater naturalness of grouping QU and CD with other classes subsumed by *some-*, we predict that QU and CD usages of *any-* will be particularly difficult for L2 speakers.

In exploring each of these hypotheses, we look for evidence in two forms: overuse of *some-* compared to native speakers, and more errors involving *any-*. We focus on the frequent semantic categories of *people* and *things*, specifically the set of IPs *someone*, *anyone*, *something*, and *anything*.[7]

3 Materials and Methods

3.1 Datasets

We expect that mastery of IPs will depend on a speaker's command of English, and therefore consider language productions both of learners (largely beginner-to-intermediate), and of L2 speakers on Reddit (shown to be highly proficient, almost on par with Reddit natives; Rabinovich et al. 2018). Our learner dataset comprises several sub-corpora: EFCAMDAT (Geertzen et al., 2013), TOEFL11 (Blanchard et al., 2013), and the freely available part of the FCE corpus (Yannakoudakis et al., 2011). The advanced L2 dataset includes online posts by advanced non-native English speakers from the L2-Reddit corpus (released by Rabinovich et al., 2018, and comprising utterances by native as well as highly-proficient non-native speakers, published on the Reddit platform). We extended the L2-Reddit corpus (originally collected in 2017) with data published through September 2018; the final dataset includes over 320M native and L2 English sentences. Table 2 presents details of the two corpora.

Dataset	Sentences	Tokens	L1s
learners	5.6M	72M	>13
advanced L2s (Reddit)	177M	2.4B	51
native (Reddit)	146M	2.1B	–

Table 2: Statistics on datasets.

[5] For this, we map our classes to Haspelmath's functions.

[6] The relative distances slightly differ, but remain highly similar across many such projections we produce.

[7] We excluded *somebody/anybody* as they are about 1/10 the frequency of their *-one* counterparts in our data.

3.2 Classification of IP Usages

Evaluating our hypotheses in Section 2 depends on assessing which usage class an utterance with a *some-/any-* pronoun belongs to, so we can compare patterns of usage and infelicities across classes. In English, the IP usage classes are often associated with particular lexical or syntactic cues in the clause with the IP – e.g., a negative adverb for DN (*I don't want anything from this collection.*), or a question mark for QU (*Would you like to buy something online?*). This enabled us to develop a rule-based classifier (see supplemental materials (A.3) for details), using a parser (Kitaev and Klein, 2018) and a set of heuristic rules.

We evaluated the classifier on sentences manually annotated by three in-house native English speakers with a background in linguistics. A sample of 750 sentences produced by Reddit native English speakers was selected for annotation, and the annotators assigned a label to each sentence from within the set of {DN, QU, CD, CP, MIXED}, where the MIXED class comprises the SP, NS, FC, and IN classes (cf. Table 1). The MIXED grouping contains classes that are (1) difficult to distinguish using simple lexical and syntactic cues (essentially, an "other" class), and (2) predicted by our linguistic analysis to be relatively similar in their error patterns. Average annotator agreement on our task was $\kappa = 0.932$; detailed annotation guidelines can be found in supplemental materials (A.2).

Table 3 shows that our rule-based classification is a reliable way to categorize a sentence with an IP (five-way classification baseline is 0.2). Because we use a subset of sentences associated with each usage class throughout our experiments, we focus on classification precision, while maintaining recall. We use this classifier to automatically label L2 sentences by usage class.

Class	DN	QU	CD	CP	MIXED
P	0.835	0.882	0.853	0.833	0.849
R	0.723	0.789	0.853	0.962	0.874
F1	0.775	0.833	0.853	0.893	0.861

Table 3: Evaluation of classification of IP usage classes.

3.3 Annotation of (In)felicitous Usages

We used the FigureEight crowdsourcing platform for collecting annotations to be used as ground truth of L2 infelicities. We extracted a randomly sampled set of 3,711 sentences from our learner corpus representing a balanced distribution over the five usage classes,[8] and a similar set of 10,000 sentences from our advanced L2 (Reddit) corpus, each containing a usage of *someone*, *something*, *anyone*, or *anything*.[9] Each sentence was annotated by five native English speakers in a choice-based annotation scheme. The occurrence of the IP in the sentence was replaced with a blank line, and each annotator marked their preference for the *some-* or *any-* pronoun in that context (or "other"), reflecting the most natural choice between the two. The gold annotation for each sentence was determined by its majority choice, and the confidence score was computed based on the number of selections (out of five annotators) of each of the two pronouns. Annotation guidelines and a sample of 500 manually annotated sentences can be found in the supplemental materials (A.4).

Table 4 presents example sentences produced by learners and L2 Reddit authors where the majority annotation unanimously differed from the original pronoun (as indicated). The utterances are provided verbatim, maintaining grammatical errors typical to productions in our corpora.

Sentences with a confidence level ≤ 0.6 are considered close to equally felicitous with either pronoun, while the confidence of 1 represents a unanimous preference for one of the alternatives. Because we used a forced-choice task, if both pronouns were acceptable (e.g., *Did you see something/anything you like?*), we expect that the confidence score will indicate the level of naturalness or typicality of the pronoun in that context. For this reason, we only consider an example infelicitous when it differs from annotator choice with a confidence ≥ 0.8, which indicates a stronger preference for one pronoun over the other.

The final annotation results include 50% (1556) and 77% (2857) of sentences with a confidence of 1.0 and of ≥ 0.8, respectively, for learners. Our advanced L2 data has 56% (5639) of sentences with a confidence of 1.0 and 81% (8079) of ≥ 0.8.

A question arises as to how meaningful it is to label an IP usage as infelicitous – i.e., the preferred IP in annotation differed from the original – if both *some-* and *any-* are in fact acceptable. To explore this, we also got crowdsourced annotations on 500

[8] Aiming at 1K per class, limited by 587 and 124 sentences in the QU and CP classes in our learner data, respectively.

[9] We excluded sentences with idiomatic expressions containing IPs from this work; see supplemental materials (A.5).

L2 utterance	Annotation
Moreover, he also takes a risk of not knowing **someone** from this country.	anyone
About 20 years ago, we didn't know **someone** who cares about them, who defend animal's right, but today, I know many people who cares about, cause animals need to be protected.	anyone
It is justified to say that they have to change **anything** to cope with the now situation.	something
I never said **something** about political science, probably it was not very good worded but my point is just that it shows how the extremes of two sides can come closer together again.	anything
I think it's a sampling bias rather than **anyone** massaging the numbers to see what they want to see.	someone
If there is a day where no one works then this is useless because you can't do **something** on that day with family besides walking in forests because everything would be closed.	anything

Table 4: Example sentences annotated by human annotators for infelicitous pronoun choice (original pronoun is boldfaced). The top part refers to learners' utterances, the bottom part refers to advanced L2s'.

native utterances from Reddit, and compared the percentages of usages annotated as infelicitous to those of 500 randomly sampled sentences by advanced L2s. We found that 3% of native utterances were annotated as infelicitous at a confidence level of ≥ 0.8, indicating a high agreement among native writers and our annotators, while for advanced L2s, the percentage was around twice that high – 6.7%. Despite acceptable variation in *some-*/*any*-usage in a given context, even advanced L2 speakers differ from natives in their relative preferences.

4 Analysis of IP Infelicities in L2

4.1 Distribution of IPs by Usage Types

First, considering **Hypothesis 1** from Section 2, we expect the confusability of *some-* and *any-* to be reflected in overgeneralization of *some-* due to its higher frequency. The subtle distinction between these pronoun types is assumed to be better mastered by advanced L2 speakers, so we expect the divergence from the native distribution to be amplified in learners' productions.

Figure 2 presents relative frequencies of *some-* and *any-* pronouns in a random sample of 5M native, advanced L2, and learner productions, both in the entire sample (left) and distributed by usage class (right). In line with our predictions, we find in Figure 2 (left) that overall, L2 speakers use *some-* pronouns more than *any-* pronouns compared to native speakers. We can further see in Figure 2 (right), and discussed in detail below, that this pattern occurs in almost all the IP usage classes, especially pronounced for learners.

Elaborating on **Hypothesis 1**, we further suggest that in addition to *general* overuse of *some-* vs. *any-* (which may partly be due to avoidance of *any-*), L2 speakers are also expected *in their infelicities* to more often use *some-* where native speakers would use *any-*, than vice versa. This prediction is also supported by our annotated data: In cases where the preferred pronoun is *some-*, learners infelicitously use *any-* 8.4% of the time, but in cases where the preferred pronoun is *any-*, learners infelicitously use *some-* almost 23% of the time. That is, learners have almost three times as many infelicities of using *some-* instead of *any-* than the reverse. Our advanced L2s speakers also show more infelicities using *some-* instead of *any-* than vice versa, but the difference is less pronounced (5.8% and 10.1% respectively), as we expect given their greater proficiency.

4.2 Distribution of Infelicitous Usages

Next we turn to **Hypothesis 2** from Section 2, which further predicts that the precise extent of deviation from native-like usage patterns will not be distributed uniformly across the different usage classes, but rather there will be a higher degree of deviation in classes that are atypically grouped under *any-* – that is, QU and CD – than in those that introduce less of a semantic challenge (DN, CP, and those in the MIXED class). L2 speakers are expected to exhibit both more overuse of *some-* and more infelicities in the QU and CD classes.

Our predictions regarding the non-uniform overuse of *some-* are largely borne out in Figure 2: the classes expected to be most difficult for L2 speakers – QU and CD – show a significant difference not only for learners, but even for advanced L2 speakers compared to natives, while DN and CP show only a difference for learners.

A few observations from Figure 2 do not follow our hypothesis. First, the difference in learner usage of *some-* vs. *any-* for DN goes in the direction opposite to the prediction: i.e., learners use *any-* more than *some-* pronouns in direct negation. We attribute this to the sheer frequency of *any-* in direct negation, such that learners are overgen-

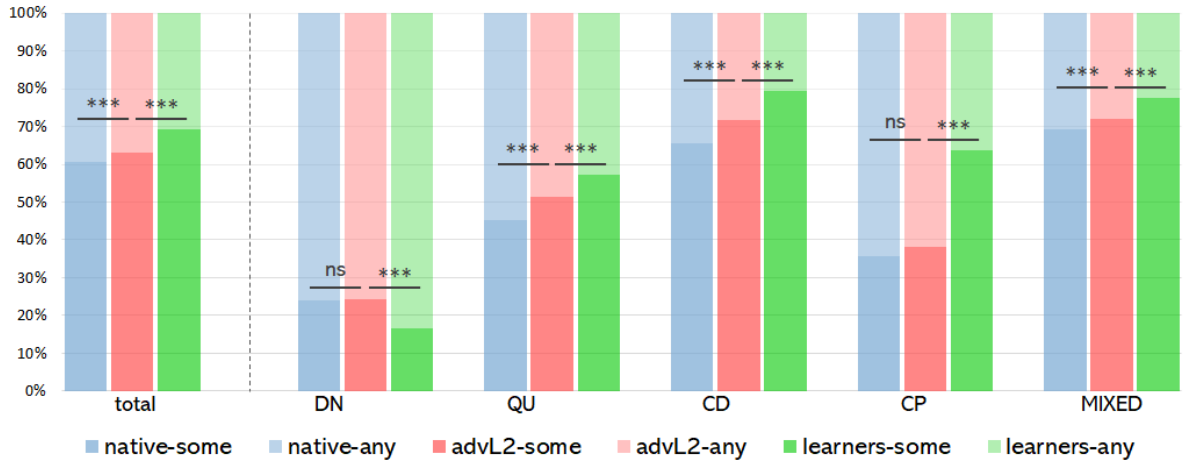


Figure 2: Distribution of *some-* and *any-* pronouns by usage class (native, advL2, learner, left-to-right in each); see Table 1 for definitions of classes. 'total' refers to *some-* and *any-* counts extracted from the sample of 5M sentences for each population. '***' indicates significant difference at the level of $p < .001$; 'ns' indicates non-significant difference.

eralizing *any-* here. Second, the MIXED grouping also shows a difference for the advanced L2 speakers, although these usages are not predicted to be especially difficult by our linguistic analysis. This class contains a very large and diverse set of usages, making it difficult to predict what is driving this effect, and we leave this for future work. Finally, the largest gap in overuse of *some-* vs. *any-* is observed in the CP class for learners, thereby not complying with our prediction of the highest difficulty being introduced by the QU and CD classes. Note, however, that this result is based on a relatively small amount of data in the CP class for learners (only 124 sentences; see Table 5).

To consider the pattern of infelicities across the usage classes, Table 5 shows the results from our crowdsourced annotation of IP usages of learners (top) and advanced L2s (bottom), separated by the classes. As expected, learners exhibit a very high percentage of infelicities in the QU class (24%); the CD class is not nearly as bad (12%), but is still higher than the other three (8–9%). Although advanced L2s have much fewer infelicities than learners, they also have more in the QU and CD classes (7% and over 9% respectively) than in the others (5–6%). Thus, as with Hypothesis 1, **Hypothesis 2** is largely borne out by the data, and we find additional evidence that the IP system of English is particularly challenging for beginning to intermediate learners.

5 Automatic Detection of Infelicities

Our motivation for the above analysis is to use these insights to drive development of tools for L2 learners. Here we consider the first step, that of detection of infelicities with a language model (LM).

Usage class	DN	QU	CD	CP	MIXED
# annotated	1000	587	1000	124	1000
# infelicitous	81	141	124	11	87
% infelicitous	8.1	24.0	12.4	8.9	8.7
# annotated	2000	2000	2000	2000	2000
# infelicitous	106	141	182	102	113
% infelicitous	5.3	7.1	9.1	5.1	5.7

Table 5: Distribution of annotated infelicities by usage class. Top panel: learners; bottom: advanced L2s.

Neural network based approaches are currently among the most successful LMs. While being easily applied to a wide range of tasks, they provide significant improvements over classic backoff n-gram models. A common use of a pre-trained LM – typically trained on an extremely large corpus – is to predict the likelihood of an 'unseen' sample of text: The higher the score (or the lower the perplexity) a text is assigned, the more probable it is, given the model. In particular, a fluent, well-formed text is likely to be scored higher by an LM than a text containing linguistic anomalies.

Encouraged by results on the task of grammatical error detection (Yuan and Briscoe, 2016; Ji et al., 2017), we adhere to a similar approach, casting the detection of infelicities as a binary classification scenario: An LM is applied on a sentence with an original pronoun (e.g., *something*) and on the same sentence where the pronoun is substituted with its alternative (e.g., *anything*); then the one predicted as more probable (scored highest) is chosen as a model decision.

5.1 Models

Aiming to test the effect of various factors, such as training data size and register, on the predictive power of LMs in our task, we used both pretrained models and models trained locally on in-domain, albeit much smaller, data.

Gulordava et al.: A successful variant of RNNs, the long short-term memory model (LSTM, Hochreiter and Schmidhuber, 1997), used for syntactic error detection in Gulordava et al. (2018). We trained the model using a similar set of parameters to Gulordava et al. (2018),[10] on 10M sentences by native English speakers of Reddit (see Section 3), using a 20K sentence validation set and a 50K sentence test set. This model allows us to test the benefits of using *in-domain* data (for advanced L2s), despite its significantly lower volume, compared to other models.

Google 1B: A very large publicly available LM released by Jozefowicz et al. (2016). This fine-tuned language model, trained on a billion-word corpus (Chelba et al., 2013), requires a massive infrastructure for training. It achieves impressive perplexity scores on common benchmarks, and has been shown effective on a range of NLP tasks.

BERT: A recent bidirectional encoder representations from transformers (BERT) LM released by Google (Devlin et al., 2018). Proven highly effective in several language modeling tasks, it achieves state-of-the-art results in syntax-sensitive scenarios (Goldberg, 2019), pushing the limits of what is feasible with current language modeling tools.

We report the models' precision, recall and F1 scores for infelicitous and correct classes separately. We also report the overall accuracy of each, computed as the ratio of correctly classified cases out of all sentences. Following the intuition laid out in Section 3.3, we conducted two sets of experiments: (1) considering cases where annotators' confidence score was 0.8 or higher, and (2) considering cases with confidence of 1. Sentences with a lower confidence score (i.e., where both *some-* and *any-* were roughly equally preferred) were excluded from these experiments.

[10] Specifically, we used two hidden layers of 200 units per layer, dropout rate of 0.2, batch size of 20, and initial learning rate of 20, and trained for 40 epochs (until the validation set perplexity converged).

5.2 Results and discussion

Tables 6 and 7 present the results for learners and advanced L2 speakers, each split by the degree of annotation confidence. Baseline accuracy is computed as the ratio of felicitous usages (the majority class) out of all instances. The Gulordava et al. LM yields results inferior to the baseline, despite training on in-domain (but much smaller) data. BERT performs best overall, and both it and Google 1B exceed the baseline for learners, but BERT performs only at baseline for advanced L2s, confirming the extreme difficulty of this task. Results obtained for the correct class are far superior to those for the infelicitous class, suggestive of the inherent difficulty of the latter cases, compared to (occasionally clear-cut) correct usage patterns.

Systematically higher scores obtained for learner utterances (Table 6), compared to advanced L2s (Table 7), imply that the mild infelicities of the latter pose a higher challenge to automatic tools. That is, not only do advanced L2s show fewer errors, but their errors are likely more subtle and more difficult to detect. The high-confidence setup ($= 1.0$) yields results superior to those produced by the lower-confidence setup (≥ 0.8), further supporting that clear-cut infelicities are more easily captured by an LM.

Returning to our linguistic predictions, the preference of *some-* over *any-* predicted by **Hypothesis 1** and shown for non-native speakers (Section 4.1) does not hold for our best-performing LM. We found a roughly equal rate (up to two percent points) of infelicities in model preferences in cases with *some-* vs. *any-* gold annotations, showing that the model (unlike non-natives) does not have greater difficulty with *any-* overall.

We also consider the non-uniform difficulty of IPs across various usage cases, predicted by **Hypothesis 2** and shown for non-natives (Section 4.2). To address this question, we test BERT for infelicitous choices compared to annotators' decisions: That is, for each sentence, we compare the pronoun preferred by the model to the gold annotation. Table 8 presents statistics across usage classes, for learners and advanced L2s (taken from Table 5), as well as for BERT. The top panel refers to learner data; the bottom panel, to advanced L2 data. While (expectedly) outperforming the two non-native populations, the model exhibits similar distributional patterns, with more infelicities in the CD and QU classes. The model also has

Learners		Infelicitous class			Correct class			
	model	P	R	F1	P	R	F1	acc
≥ 0.8	Gulordava et al. (trained on Reddit)	0.437	0.573	0.496	0.920	0.870	0.894	0.825
	Google 1B (pre-trained)	0.500	0.686	0.578	0.946	0.889	0.917	0.861
	BERT (pre-trained)	**0.602**	**0.736**	**0.673**	**0.956**	**0.911**	**0.933**	**0.889**
$= 1$	Gulordava et al. (trained on Reddit)	0.499	0.652	0.565	0.954	0.916	0.935	0.887
	Google 1B (pre-trained)	0.523	0.720	0.606	0.970	0.932	0.950	0.912
	BERT (pre-trained)	**0.681**	**0.859**	**0.759**	**0.981**	**0.949**	**0.965**	**0.939**

Table 6: Automatic detection of infelicities in learner data (sentences where annotation disagrees with author usage of IP), with confidence level ≥ 0.8 (top), and with confidence level $= 1$ (bottom). Baseline accuracy is 0.850 for the former and 0.887 for the latter. Best result in a column (for each part) is boldfaced.

Advanced L2s		Infelicitous class			Correct class			
	model	P	R	F1	P	R	F1	acc
≥ 0.8	Gulordava et al. (trained on Reddit)	0.274	0.583	0.373	0.959	0.863	0.908	0.840
	Google 1B (pre-trained)	0.380	0.704	0.494	0.976	0.912	0.943	0.898
	BERT (pre-trained)	**0.506**	**0.701**	**0.585**	**0.972**	**0.938**	**0.955**	**0.919**
$= 1$	Gulordava et al. (trained on Reddit)	0.219	0.690	0.332	0.984	0.886	0.932	0.877
	Google 1B (pre-trained)	0.380	0.760	0.507	0.988	0.942	0.964	0.934
	BERT (pre-trained)	**0.503**	**0.790**	**0.614**	**0.990**	**0.964**	**0.977**	**0.956**

Table 7: Automatic detection of infelicities in advanced L2 data (sentences where annotation disagrees with author usage of IP), with confidence level ≥ 0.8 (top), and with confidence level $= 1$ (bottom). Baseline accuracy is 0.918 for the former and 0.956 for the latter. Best result in a column (for each part) is boldfaced.

a higher number of infelicities in the CP class for learners; again, we note the small sample of data in this class, entailing a need for further investigation of this particular pattern. The model results here pose intriguing questions for future work regarding the nature of challenges faced by automatic neural methods, and their potential analogues to those of humans.

	DN	QU	CD	CP	MIXED
learners	8.1	24.0	12.4	8.9	8.7
BERT	0.8	6.1	3.6	4.0	2.2
advanced L2s	5.3	7.1	9.1	5.1	5.7
BERT	1.3	2.5	2.7	1.6	1.5

Table 8: Distribution of % of infelicities (difference from gold annotation) across classes for humans and for BERT on the corresponding data.

6 Related Work

Computational approaches to grammatical error correction (GEC) in learners' productions has been a prolific field of research in recent years. A standard approach to dealing with grammar and spelling errors makes use of a machine-learning classification paradigm; a comprehensive survey of these methods can be found in Ng et al. (2014). Recent advances in the field of GEC were achieved by using neural models (Yuan and Briscoe, 2016; Ji et al., 2017; Sakaguchi et al., 2017; Lo et al., 2018). Most studies used a *supervised* setup for selecting a correct choice (e.g., a preposition) out of a set of multiple alternatives, rendering our experimental setup not directly comparable.

Another line of work has assessed the capability of neural LMs to capture errors stemming from violation of syntax-sensitive dependencies (Linzen et al., 2016; Gulordava et al., 2018; Marvin and Linzen, 2018). The recent BERT model (Devlin et al., 2018) has been shown to be highly effective for detection of syntactic anomalies stemming from subject-verb disagreement (Goldberg, 2019).

Most research on L2 error correction focuses on function words, such as prepositions and determiners. Very little work has been done on detecting and correcting incorrect usage of content words. Most has been focused on the felicity of word combinations, such as identifying disfluencies stemming from L1 paraphrases (e.g., *eat medicine* or *look movies*, Brooke and Hirst, 2011; Dahlmeier and Ng, 2011), or using models of compositionality to detect semantically deviant pairs (*residential steak*, Vecchi et al., 2011) or infelicitous collocations (?*big importance* vs. *great importance*, Kochmar and Briscoe, 2013). A shared task on automatic evaluation of scientific writing (Daudaravicius et al., 2016) addressed automatic detection of a variety of grammatical errors (e.g., misuse of an article or punctuation) and lexical infelicities (e.g., phrasing choices stem-

ming from style requirements of the genre) in scientific papers, edited by a professional company.

While most closely related to the field of semantic error detection, our work deals with subtle linguistic choices that shape the ultimate attainment of L2 in non-native speakers. Compared to grammatical and semantic anomalies explored in previous work, the choice of indefinite pronoun is often guided by implicit contextual clues that are not necessarily reflected in superficial collocational patterns, thereby posing a higher challenge for automatic techniques.

7 Conclusion

We develop and evaluate linguistic hypotheses on the difficulties for second language learners of the atypical system of English indefinite pronouns. We find that the tangled relation between *some-* and *any-* pronouns pose challenges that are evident in the productions of both learners and advanced L2 speakers. This work thus demonstrates the promise of extending computational approaches for error-detection in L2 productions to more subtle semantic usages. Moreover, our results reveal the challenges that these subtleties can pose for even advanced non-native speakers.

Much research in second language acquisition establishes *native language transfer* as one of the major factors that shape productions of non-native speakers. While the work here addresses *universal* (i.e., native-language independent) challenges posed to L2 speakers, a plausible assumption is that mastery of English IPs is also affected by the proximity of the analogous system in a speaker's L1. We leave this direction for future research.

We also evaluate here the ability of language models to detect the errors arising in the use of English indefinite pronouns in L2 productions. Not surprisingly, we find that the more clearcut errors exhibited by learners are easier to automatically identify than the potentially more subtle errors that arise with advanced L2 speakers. The best performing language model shows a varying match to human patterns of difficulty, raising issues for further research regarding the factors that influence difficulty for both humans and language models.

The practical impact of this work will be in facilitating the development of educational applications for L2 English speakers at various levels of proficiency. At present, most error correction and detection tools focus on explicit spelling or grammar errors. Enriching these tools with the ability to capture subtle semantic infelicities in the usage of IPs would advance the current state of the art in educational applications for language learners.

Acknowledgments

This research is supported by an NSERC Discovery Grant RGPIN-2017-06506 to Suzanne Stevenson. We are thankful to Paola Merlo for her insight and advice. We are also grateful to our anonymous reviewers for their constructive feedback.

References

Lloyd B Anderson. 1982. The 'perfect' as a universal and as a language-specific category. In Paul J. Hopper, editor, *Tense-aspect: Between semantics and pragmatics*, pages 227–264. John Benjamins, Amsterdam.

Barend Beekhuizen, Julia Watson, and Suzanne Stevenson. 2017. Semantic typology and parallel corpora: Something about indefinite pronouns. In *Proceedings of the 39th Annual Conference of the Cognitive Science Society*.

Brent Berlin and Paul Kay. 1969. *Basic color terms: Their university and evolution*. California UP.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A corpus of non-native English. *ETS Research Report Series*, 2013(2):i–15.

Julian Brooke and Graeme Hirst. 2011. Lexicalizing computational stylistics for language learner feedback. In *Proceedings, Conference on Stylistics Across Disciplines*, Leiden.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report.

Daniel Dahlmeier and Hwee Tou Ng. 2011. Correcting semantic collocation errors with l1-induced paraphrases. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 107–117. Association for Computational Linguistics.

Vidas Daudaravicius, Rafael E Banchs, Elena Volodina, and Courtney Napoles. 2016. A report on the automatic evaluation of scientific writing shared task. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 53–62.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of

deep bidirectional transformers for language understanding. Technical Report arXiv:1810.04805 [cs.CL].

Alexandre Franois. 2008. Semantic maps and the typology of colexification: intertwining polysemous networks across languages. In Martine Vanhove, editor, *From polysemy to semantic change*, pages 163–215. Benjamins, Amsterdam.

Jeroen Geertzen, Theodora Alexopoulou, and Anna Korhonen. 2013. Automatic linguistic annotation of large scale L2 databases: The EF-Cambridge open language database (EFCAMDAT). In *Proceedings of the 31st Second Language Research Forum. Somerville, MA: Cascadilla Proceedings Project*.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. Technical Report arXiv:1901.05287 [cs.CL].

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1195–1205. Association for Computational Linguistics.

Martin Haspelmath. 1997. *Indefinite pronouns*. Oxford University Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 753–762.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. Technical Report arXiv:1602.02410 [cs.CL].

Nikita Kitaev and Dan Klein. 2018. Multilingual constituency parsing with self-attention and pre-training. Technical Report arXiv:1812.11760 [cs.CL].

Ekaterina Kochmar and Ted Briscoe. 2013. Capturing anomalies in the choice of content words in compositional distributional semantic space. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 365–372.

Stephen Levinson, Sérgio Meira, The Language, and Cognition Group. 2003. 'natural concepts' in the spatial topological domain-adpositional meanings in crosslinguistic perspective: An exercise in semantic typology. *Language*, pages 485–516.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association of Computational Linguistics*, 4(1):521–535.

Yu-Chun Lo, Jhih-Jie Chen, Chingyu Yang, and Jason Chang. 2018. Cool english: A grammatical error correction system based on large learner corpora. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 82–85.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14.

Ella Rabinovich, Yulia Tsvetkov, and Shuly Wintner. 2018. Native language cognate effects on second language lexical choice. *Transactions of the Association of Computational Linguistics*, 6:329–342.

Alla Rozovskaya, Dan Roth, and Mark Sammons. 2017. Adapting to learner errors with minimal supervision. *Computational Linguistics*, 43(4):723–760.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, volume 2, pages 366–372.

Eva Maria Vecchi, Marco Baroni, and Roberto Zamparelli. 2011. (linear) maps of the impossible: capturing semantic anomalies in distributional space. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pages 1–9. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Compositional Generalization in Image Captioning

Mitja Nikolaus
University of Tübingen*
mitja.nikolaus@posteo.de

Mostafa Abdou
University of Copenhagen
abdou@di.ku.dk

Matthew Lamm
Stanford University
mlamm@cs.stanford.edu

Rahul Aralikatte
University of Copenhagen
rahul@di.ku.dk

Desmond Elliott
University of Copenhagen
de@di.ku.dk

Abstract

Image captioning models are usually evaluated on their ability to describe a held-out set of images, not on their ability to generalize to unseen concepts. We study the problem of compositional generalization, which measures how well a model composes unseen combinations of concepts when describing images. State-of-the-art image captioning models show poor generalization performance on this task. We propose a multi-task model to address the poor performance, that combines caption generation and image–sentence ranking, and uses a decoding mechanism that re-ranks the captions according their similarity to the image. This model is substantially better at generalizing to unseen combinations of concepts compared to state-of-the-art captioning models.

1 Introduction

When describing scenes, humans are able to almost arbitrarily combine concepts, producing novel combinations that they have not previously observed (Matthei, 1982; Piantadosi and Aslin, 2016). Imagine encountering a purple-colored dog in your town, for instance. Given that you understand the concepts PURPLE and DOG, you are able to compose them together to describe the dog in front of you, despite never having seen one before.

Image captioning models attempt to automatically describe scenes in natural language (Bernardi et al., 2016). Most recent approaches generate captions using a recurrent neural network, where the image is represented by features extracted from a Convolutional Neural Network (CNN). Although state-of-the-art models show good performance on challenge datasets, as measured by text-similarity metrics, their performance as measured by human judges is low when compared to human-written captions (Vinyals et al., 2017, Section 5.3.2).

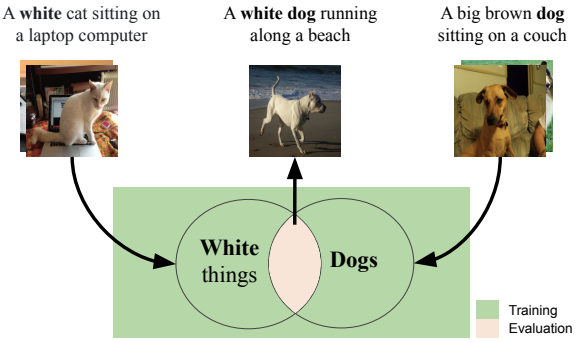


Figure 1: We evaluate whether image captioning models are able to compositionally generalize to unseen combinations of adjectives, nouns, and verbs by forcing *paradigmatic gaps* in the training data.

It is widely believed that systematic compositionality is a key property of human language that is essential for making generalizations from limited data (Montague, 1974; Partee, 1984; Lake et al., 2017). In this work, we investigate to what extent image captioning models are capable of compositional language understanding. We explore whether these models can compositionally generalize to unseen adjective–noun and noun–verb composition pairs, in which the constituents of the pair are observed during training but the combination is not, thus introducing a *paradigmatic gap* in the training data, as illustrated in Figure 1. We define new training and evaluation splits of the COCO dataset (Chen et al., 2015) by holding out the data associated with the compositional pairs from the training set. These splits are used to evaluate how well models generalize to describing images that depict the held out pairings.

We find that state-of-the-art captioning models, such as Show, Attend and Tell (Xu et al., 2015), and Bottom-Up and Top-Down Attention (Ander-

*The work was carried out during a visit to the University of Copenhagen.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 87–98
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

son et al., 2018), have poor compositional generalization performance. We also observe that the inability to generalize of these models is primarily due to the language generation component, which relies too heavily on the distributional characteristics of the dataset and assigns low probabilities to unseen combinations of concepts in the evaluation data. This supports the findings from concurrent work (Holtzman et al., 2019) which studies the challenges in decoding from language models trained with a maximum likelihood objective.

To address the generalization problem, we propose a multi-task model that jointly learns image captioning and image–sentence ranking. For caption generation, our model benefits from an additional step, where the set of captions generated by the model can be re-ranked using the jointly-trained image–sentence ranking component. We find that the ranking component is less affected by the likelihood of n-gram sequences in the training data, and that it is able to assign a higher ranking to more informative captions which contain unseen combinations of concepts. These findings are reflected by improved compositional generalization.

The source code is publicly available on GitHub.[1]

2 Related Work

2.1 Caption Generation and Retrieval

Image Caption Generation models are usually end-to-end differentiable encoder-decoder models trained with a maximum likelihood objective. Given an image encoding that is extracted from a convolutional neural network (CNN), an RNN-based decoder generates a sequence of words that form the corresponding caption (Vinyals et al., 2015, *inter-alia*). This approach has been improved by applying top-down (Xu et al., 2015) and bottom-up attention mechanisms (Anderson et al., 2018). These models show increasingly good performance on benchmark datasets, e.g. COCO, and in some cases reportedly surpass human-level performance as measured by n-gram based evaluation metrics (Bernardi et al., 2016). However, recent work has revealed several caveats. Firstly, when using human judgments for evaluation, the automatically generated captions are still considered worse in most cases (Fang et al., 2015; Vinyals et al., 2017). Furthermore, when evaluating out-of-domain images or images with unseen concepts, it has been shown that the generated captions are often of poor quality (Mao et al., 2015; Vinyals et al., 2017). Attempts have been made to address the latter issue by leveraging unpaired text data or pre-trained language models (Hendricks et al., 2016; Agrawal et al., 2018).

Image–Sentence Ranking is closely related to image captioning. Here, the problem of language generation is circumvented and models are instead trained to rank a set of captions given an image, and vice-versa (Hodosh et al., 2013). A common approach is to learn a visual–semantic embedding for the captions and images, and to rank the images or captions based on similarity in the joint embedding space. State-of-the-art models extract image features from CNNs and use gated RNNs to represent captions, both of which are projected into a joint space using a linear transformation (Frome et al., 2013; Karpathy and Fei-Fei, 2015; Vendrov et al., 2016; Faghri et al., 2018).

2.2 Compositional Models of Language

Investigations of compositionality in vector space models date back to early debates in the cognitive science (Fodor and Pylyshyn, 1988; Fodor and Lepore, 2002) and connectionist literature (McClelland et al., 1986; Smolensky, 1988) regarding the ability of connectionist systems to compose simple constituents into complex structures. In the NLP literature, numerous approaches that (loosely) follow the linguistic principle of compositionality[2] have been proposed (Mitchell and Lapata, 2008; Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011). More recently, it has become standard to employ representations which are learned using neural network architectures. The extent to which these models behave compositionally is an open topic of research (Lake and Baroni, 2017; Dasgupta et al., 2018; Ettinger et al., 2018; McCoy et al., 2018) that closely relates to the focus of the present paper.

Compositional generalization in image captioning has received limited attention in the literature. In Atzmon et al. (2016), the captions in the COCO dataset are replaced by subject-relation-object triplets, circumventing the problem of language generation, and replacing it with structured

[1] https://github.com/mitjanikolaus/compositional-image-captioning

[2] *The meanings for complex expressions are derived from the meanings of their parts via specific composition functions.* (Partee, 1984)

triplet prediction. Other work explores generalization to unseen combinations of visual concepts as a classification task (Misra et al., 2017; Kato et al., 2018). Lu et al. (2018) is more closely related to our work; they evaluate captioning models on describing images with unseen noun-noun pairs.

In this paper, we study compositional generalization in image captioning with combinations of multiple classes of nouns, adjectives, and verbs.[3] We find that state-of-the-art models fail to generalize to unseen combinations, and present a multi-task model that improves generalization by combining image captioning (Anderson et al., 2018) and image–sentence ranking (Faghri et al., 2018). In contrast to other models that use a re-ranking step[4], our model is trained jointly on both tasks and does not use any additional features or external resources. The ranking model is only used to optimize the global semantics of the generated captions with respect to the image.

3 Compositional Image Captioning

3.1 Problem Definition

In this section we define the compositional captioning task, which is designed to evaluate how well a model generalizes to captioning images that *should* be described using previously unseen combinations of concepts, when the individual concepts have been observed in the training data.

We assume a dataset of captioned images $\mathcal{D}$, in which N images are described by K captions: $\mathcal{D} := \{\langle i^1, s_1^1, ..., s_K^1\rangle, ..., \langle i^N, s_1^N, ..., s_K^N\rangle\}$. We also assume the existence of a concept pair $\{c_i, c_j\}$ that represents the concepts of interest in the evaluation. In order to evaluate the compositional generalization of a model for that concept pair, we first define a training set by identifying and removing instances where the captions of an image contain the pair of concepts, creating a *paradigmatic gap* in the original training set: $\mathcal{D}_{\text{train}} := \{\langle i^n, s_k^n\rangle\}$ s.t. $\forall_{n=1}^{N} \nexists\ k : c_i \in s_k^n \wedge c_j \in s_k^n$. Note that the concepts c_i and c_j can still be independently observed in the captions of an image of this set, but *not* together in the same caption. We also define validation and evaluation sets D_{val} and D_{eval} that *only* contain instances where at least one of the captions of an image contains the pair of concepts: $\mathcal{D}_{\text{val/eval}} := \{\langle i^n, s_k^n\rangle\}$ s.t. $\forall_{n=1}^{N} \exists\ k : c_i \in s_k^n \wedge c_j \in s_k^n$. A model is trained on the $\mathcal{D}_{\text{train}}$ training set until it converges, as measured on the $\mathcal{D}_{\text{val}}$ validation set. The compositional generalization of the model is measured by the proportion of evaluation set captions which successfully combined a held out pair of concepts $\{c_i, c_j\}$ in D_{eval}.

3.2 Selection of Concept Pairs

We select pairs of concepts that are likely to be represented in an image recognition model. In particular, we identify adjectives, nouns, and verbs in the English COCO captions dataset (Chen et al., 2015) that are suitable for testing compositional generalization. We define concepts as sets of synonyms for each word, to account for the variation in how the concept can be expressed in a caption. For each noun, we use the synonyms defined in Lu et al. (2018). For the verbs and adjectives, we use manually defined synonyms (see Appendix D). From these concepts, we select adjective–noun and noun–verb pairs for the evaluation. To identify concept pair candidates, we use StanfordNLP (Qi et al., 2018) to label and lemmatize the nouns, adjectives, and verbs in the captions, and to check if the adjective or verb is connected to the respective noun in the dependency parse.

Nouns: We consider the 80 COCO object categories (Lin et al., 2014) and additionally divide the "person" category into "man", "woman" and "child". It has been shown that models can detect and classify these categories with high confidence (He et al., 2016). We further group the nouns under consideration into animate and inanimate objects. We use the following nouns in the evaluation: `woman`, `man`, `dog`, `cat`, `horse`, `bird`, `child`, `bus`, `plane`, `truck`, `table`.

Adjectives: We analyze the distribution of the adjectives in the dataset (see Figure 4 in Appendix A). The captions most frequently contain descriptions of the color, size, age, texture or quantity of objects in the images. We consider the color and size adjectives in this evaluation. It has been shown that CNNs can accurately classify the color of objects (Anderson et al., 2016); and we assume that CNNs can encode the size of objects because they can predict bounding boxes, even for small

[3] This is different from the "robust image captioning" task (Lu et al., 2018) because we are testing for the composition of nouns with adjectives or verbs, and not the co-occurrence of different nouns in an image.

[4] Fang et al. (2015) use a discriminative model that has access to sentence-level features and a multimodal similarity model in order to capture global semantics. Wang et al. (2017) uses a conditional variational auto-encoder to generate a set of diverse captions and a consensus-based method for re-ranking the candidates.

black cat	big bird	red bus
small plane	eat man	lie woman
white truck	small cat	brown dog
big plane	ride woman	fly bird
white horse	big cat	blue bus
small table	hold child	stand bird
black bird	small dog	white boat
stand child	big truck	eat horse

Table 1: The 24 concept pairs used to construct the training $\mathcal{D}_{\text{train}}$ and eval $\mathcal{D}_{\text{eval}}$ datasets.

objects (Bai et al., 2018). In the evaluation, we use the following adjectives: `big`, `small`, `black`, `red`, `brown`, `white`, `blue`.

Verbs: Sadeghi and Farhadi (2011) show that it is possible to automatically describe the interaction of objects or the activities of objects in images. We select verbs that describe simple and well-defined actions and group them into transitive and intransitive verbs. We use the following verbs in the pairs: `eat`, `lie`, `ride`, `fly`, `hold`, `stand`.

Pairs and Datasets: We define a total of 24 concept pairs for the evaluation, as shown in Table 1. The training and evaluation data is extracted from the COCO dataset, which contains K=5 reference captions for N=123,287 images. In the compositional captioning evaluation, we define the training datasets D_{train} and validation datasets D_{val} as subsets of the original COCO training data, and the evaluation datasets D_{eval} as subsets of the COCO validation set, both given the concept pairs.

To ensure that there is enough evaluation data, we only use concept pairs for which there are more than 100 instances in the validation set. Occurrence statistics for the considered concept pairs can be found in Appendix B.

3.3 Evaluation Metric

The performance of a model is measured on the $\mathcal{D}_{\text{eval}}$ datasets. For each concept pair evaluation set consisting of M images, we dependency parse the set of $M \times K$ generated captions $\{\langle s_1^1, ..., s_K^1\rangle, ..., \langle s_1^M, ..., s_K^M\rangle\}$ to determine whether the captions contain the expected concept pair, and whether the adjective or verb is a dependent of the noun.[5] We denote the set of captions for which these conditions hold true as $\mathcal{C}$.

[5]This means that a model gains no credit for predicting the concept pairs without them attaching to their expected target.

There is low inter-annotator agreement in the human reference captions on the usage of the concepts in the target pairs.[6] Therefore, one should *not* expect a model to generate a *single* caption with the concepts in a pair. However, a model can generate a larger set of K captions using beam search or diverse decoding strategies. Given K captions, the recall of the concept pairs in an evaluation dataset is:

$$\text{Recall@K} = \frac{|\{\langle s_k^m\rangle \mid \exists k : s_k^m \in \mathcal{C}\}|}{M} \quad (1)$$

Recall@K is an appropriate metric because the reference captions were produced by annotators who did not need to produce any specific word when describing an image. In addition, the set of captions $\mathcal{C}$ is determined with respect to the same synonym sets of the concepts that were used to construct the datasets, and so credit is given for semantically equivalent outputs. More exhaustive approaches to determine semantic equivalence for this metric are left for future work.

4 State-of-the-Art Performance

4.1 Experimental Protocol

Models: We evaluate two image captioning models on the compositional generalization task: Show, Attend and Tell (SAT; Xu et al., 2015) and Bottom-up and Top-down Attention (BUTD; Anderson et al., 2018). For SAT, we use ResNet-152 (He et al., 2016) as an improved image encoder.

Training and Evaluation: The models are trained on the $\mathcal{D}_{\text{train}}$ datasets, in which groups of concept pairs are held out—see Appendix C for more information. Hyperparameters are set as described in the respective papers. When a model has converged on the $\mathcal{D}_{\text{val}}$ validation split (as measured in BLEU score), we generate K captions for each image in $\mathcal{D}_{\text{eval}}$ using beam search. Then, we calculate the Recall@K metric (Eqn. 1, K=5) for each concept pair in the evaluation split, as well as the average over all recall scores to report the compositional generalization performance of a model.

We also evaluate the compositional generalization of a BUTD model trained on the full COCO

[6]We calculate the inter-annotator agreement for the target pairs between the 5 reference captions for every image in the COCO dataset: on average, only 1.57 / 5 captions contain the respective adjective–noun or noun–verb concept pair, if it is present in any. We ascribe this lack of agreement to the open nature of the annotation task: there were no restrictions given for what should be included in an image caption.

training dataset (FULL). In this setting, the model is trained on compositions of the type we seek to evaluate in this task, and thus does not need to generalize to new compositions.

Pretrained Language Representations: The word embeddings of image captioning models are usually learned from scratch, without pretraining[7]. Pretrained word embeddings (e.g. GloVe (Pennington et al., 2014)) or language models (e.g. Devlin et al. (2019)) contain distributional information obtained from large-scale textual resources, which may improve generalization performance. However, we do use them for this task because the resulting model may not have the expected paradigmatic gaps.

4.2 Results

Image Captioning: The models mostly fail to generate captions that contain the held out pairs. The average Recall@5 for SAT and BUTD are 3.0 and 6.5, respectively. A qualitative analysis of the generated captions shows that the models usually describe the depicted objects correctly, but, in the case of held out adjective–noun pairs, the models either avoid using adjectives, or use adjectives that describe a different property of the object in question, e.g. *white and green airplane* instead of *small plane* in Figure 3. In the case of held out noun–verb pairs, the models either replace the target verb with a less descriptive phrase, e.g. *a man sitting with a plate of food* instead of *a man is eating* in Figure 3, or completely omit the verb, reducing the caption to a simple noun phrase.

In the FULL setting, average Recall@5 reaches 33.3. We assume that this score is a conservative estimate due to the low average inter-annotator agreement (see Footnote 6). The model is less likely to describe an image using the target pair if the pair is only present in one of the reference captions, as the feature is likely not salient (e.g. the car in the image has multiple colors, and the target color is only covering one part of the car). In fact, if we calculate the average recall for images where at least 2 / 3 / 4 / 5 of the reference captions contain the target concept pair, Recall@5 increases to 46.5 / 58.3 / 64.9 / 75.2. This shows that the BUTD model is more likely to generate a caption with the expected concept pair when more human annotators agree that it is a salient pair of concepts in an image.

[7]Exceptions: You et al. (2016); Anderson et al. (2017)

Image–Sentence Ranking: In a related experiment, we evaluate the generalization performance of the VSE++ image–sentence ranking model on the compositional captioning task (Faghri et al., 2018). We use an adapted version of the evaluation metric because the ranking model does not generate tokens.[8] The average Recall@5 with the adapted metric for the ranking model is 46.3. The respective FULL performance for this model is 47.0, indicating that the model performs well whether it has seen examples of the evaluation concept pair at training time or not. In other words, the model achieves better compositional generalization than the captioning models.

5 Joint Model

In the previous section, we found that state-of-the-art captioning models fail to generalize to unseen combinations of concepts, however, an image-sentence ranking model does generalize. We propose a multi-task model that is trained for image captioning and image–sentence ranking with shared parameters between the different tasks. The captioning component can use the ranking component to re-rank complete candidate captions in the beam. This ensures that the generated captions are as informative and accurate as possible, given the constraints of satisfying both tasks.

Following Anderson et al. (2018), the model is a two-layer LSTM (Hochreiter and Schmidhuber, 1997), where the first layer encodes the sequence of words, and the second layer integrates visual features from the bottom-up and top-down attention mechanism, and generates the output sequence. The parameters of the ranking component θ_2 are mostly a subset of the parameters of the generation component θ_1. We name the model **B**ottom-**U**p and **T**op-down attention with **R**anking (BUTR). Figure 2 shows a high-level overview of the model architecture.

5.1 Image–Sentence Ranking

To perform the image–sentence ranking task, we project the images and captions into a joint visual-semantic embedding space $\mathbb{R}^J$. We introduce a

[8]For each image in the evaluation set, we construct a test set that consists of the 5 correct captions and the captions of 1,000 randomly selected images from the COCO validation set. We ensure that all captions in the test set contain exactly one of the constituent concept pairs, but not both (except for the 5 correct captions). We construct a ranking of the captions in this test set with respect to the image, and use the top-K ranked captions to calculate the concept pair recall (Eqn. 1).

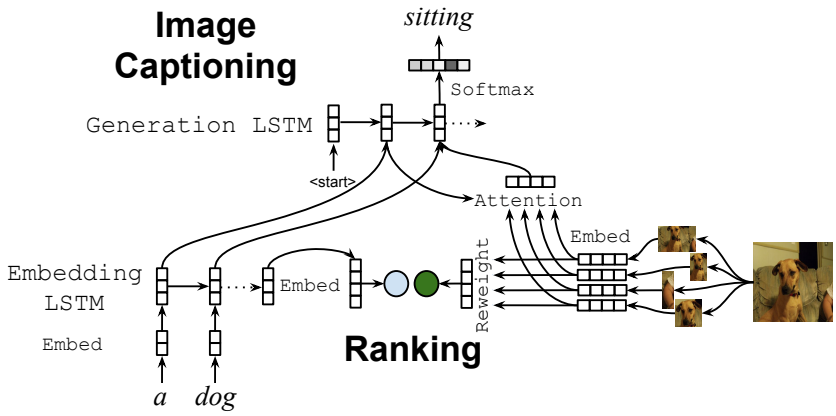


Figure 2: An overview of BUTR, which jointly learns image–sentence ranking and image captioning.

language encoding LSTM with a hidden layer dimension of L.

$$\boldsymbol{h}_t^l = \text{LSTM}(\boldsymbol{W}_1 \boldsymbol{o}_t, \boldsymbol{h}_{t-1}^l) \tag{2}$$

where $\boldsymbol{o}_t \in \mathbb{R}^V$ is a one-hot encoding of the input word at timestep t, $\boldsymbol{W}_1 \in \mathbb{R}^{E \times V}$ is a word embedding matrix for a vocabulary of size V and $\boldsymbol{h}_{t-1}^l$ the state of the LSTM at the previous timestep. At training time, the input words are the words of the target caption at each timestep.

The final hidden state of the language encoding LSTM $\boldsymbol{h}_{t=T}^l$ is projected into the joint embedding space as $\boldsymbol{s}^* \in \mathbb{R}^J$ using $\boldsymbol{W}_2 \in \mathbb{R}^{J \times L}$:

$$\boldsymbol{s}^* = \boldsymbol{W}_2 \boldsymbol{h}_{t=T}^l \tag{3}$$

The images are represented using the bottom-up features proposed by Anderson et al. (2018). For each image, we extract a set of R mean-pooled convolutional features $v_r \in \mathbb{R}^I$, one for each proposed image region r. We introduce $\boldsymbol{W}_3 \in \mathbb{R}^{J \times I}$, which projects the image features of a single region into the joint embedding space:

$$\boldsymbol{v}_r^e = \boldsymbol{W}_3 \boldsymbol{v}_r \tag{4}$$

To form a single representation $\boldsymbol{v}^*$ of the image from the set of embedded image region features $\boldsymbol{v}_r^e$, we apply a weighting mechanism. We generate a normalized weighting of region features $\beta \in \mathbb{R}^R$ using $\boldsymbol{W}_4 \in \mathbb{R}^{1 \times J}$. β_r denotes the weight for a specific region r. Then we sum the weighted region features to generate $\boldsymbol{v}^* \in \mathbb{R}^J$:

$$\beta_r' = \boldsymbol{W}_4 \boldsymbol{v}_r^e \tag{5}$$

$$\boldsymbol{\beta} = \text{softmax}(\boldsymbol{\beta}') \tag{6}$$

$$\boldsymbol{v}^* = \sum_{r=1}^{R} \beta_r \boldsymbol{v}_r^e \tag{7}$$

We define the similarity between an image and a caption as the cosine similarity $cos(\boldsymbol{v}^*, \boldsymbol{s}^*)$.

5.2 Caption Generation

For caption generation, we introduce a separate language generation LSTM that is stacked on top of the language encoding LSTM. At each timestep t, we first calculate a weighted representation of the input image features. We calculate a normalized attention weight $\alpha_t \in \mathbb{R}^R$ (one $\alpha_{r,t}$ for each region) using the language encoding and the image region features. Then, we create a single weighted image feature vector:

$$\alpha_{r,t}' = \boldsymbol{W}_5 \tanh(\boldsymbol{W}_6 \boldsymbol{v}_r^e + \boldsymbol{W}_7 h_t^l) \tag{8}$$

$$\boldsymbol{\alpha}_t = \text{softmax}(\boldsymbol{\alpha}_{r,t}') \tag{9}$$

$$\hat{\boldsymbol{v}}_t = \sum_{r=1}^{R} \boldsymbol{\alpha}_{r,t} \boldsymbol{v}_r^e \tag{10}$$

where $\boldsymbol{W}_5 \in \mathbb{R}^H$, $\boldsymbol{W}_6 \in \mathbb{R}^{H \times J}$ and $\boldsymbol{W}_7 \in \mathbb{R}^{H \times L}$. H indicates the hidden layer dimension of the attention module.

These weighted image features $\hat{\boldsymbol{v}}_t$, the output of the language encoding LSTM $\boldsymbol{h}_t^l$ (Eqn. 2) and the previous state of the language generation LSTM $\boldsymbol{h}_{t-1}^g$ are input to the language generation LSTM:

$$\boldsymbol{h}_t^g = \text{LSTM}([\hat{\boldsymbol{v}}_t, \boldsymbol{h}_t^l], \boldsymbol{h}_{t-1}^g) \tag{11}$$

The hidden layer dimension of the LSTM is G. The output probability distribution over the vocabulary is calculated using $\boldsymbol{W}_8 \in \mathbb{R}^{V \times G}$:

$$p(w_t | w_{<t}) = \text{softmax}(\boldsymbol{W}_8 \boldsymbol{h}_t^g) \tag{12}$$

5.3 Training

The model is jointly trained on two objectives. The caption generation component is trained with

a cross-entropy loss, given a target ground-truth sentence s consisting of the words $w_1, \ldots, w_T$:

$$\mathcal{L}_{\text{gen}}(\theta_1) = -\sum_{t=1}^{T} \log p(w_t | w_{<t}; i) \quad (13)$$

The image–caption ranking component is trained using a hinge loss with emphasis on hard negatives (Faghri et al., 2018):

$$\mathcal{L}_{\text{rank}}(\theta_2) = \max_{s'}[\alpha + \cos(i, s') - \cos(i, s)]_+ + \max_{i'}[\alpha + \cos(i', s) - \cos(i, s)]_+ \quad (14)$$

where $[x]_+ \equiv max(x, 0)$.

These two loss terms can take very different magnitudes during training, and thus can not be simply added. We use GradNorm (Chen et al., 2018) to learn loss weighting parameters w_{gen} and w_{rank} with an additional optimizer during training. These parameters dynamically rescale the gradients so that no task becomes too dominant. The overall training objective is formulated as the weighted sum of the single-task losses:

$$\mathcal{L}(\theta_1, \theta_2) = w_{\text{gen}}\mathcal{L}_{\text{gen}}(\theta_1) + w_{\text{rank}}\mathcal{L}_{\text{rank}}(\theta_2) \quad (15)$$

5.4 Inference

The model generates B captions for each image using beam search decoding. At each timestep, the tokens generated so far for each item on the beam are input back into the language encoder (Eqn. 3). The output of the language encoder is concatenated with the image representation (Eqn. 7) and the previous hidden state of the generation LSTM, and input to the generation LSTM (Eqn. 11) to predict the next token (Eqn. 12).

The jointly-trained image–sentence ranking component can be used to re-rank the generated captions comparing the image embedding with a language encoder embedding of the captions (Eqn. 4). We expect the ranking model will produce a better ranking of the B captions than only beam search by considering their relevance and informativity with respect to the image.

6 Results

We follow the experimental protocol defined in Section 4 to evaluate the joint model. See Appendix E for training details and hyperparameters.

Table 2 shows the compositional generalization performance, as well as the common image captioning metric scores for all models. BUTR uses the same image features and a decoder architecture as the BUTD model. Thus, when using the standard beam search decoding method, BUTR does not improve over BUTD. However, when using the improved decoding mechanism with re-ranking BUTR + RR, Recall@5 increases to 13.2. We also observe an improvement in METEOR and SPICE, and a drop in BLEU and CIDEr compared to the other models. We note that BLEU has the weakest correlations (Elliott and Keller, 2014), and SPICE and METEOR have the strongest correlations with human judgments (Kilickaya et al., 2017).

Model	R	M	S	C	B
SAT	3.0	23.2	16.6	80.4	27.5
BUTD	6.5	25.8	19.1	**98.1**	**32.6**
BUTR	6.5	25.7	19.0	97.0	32.0
BUTR + RR	**13.2**	**26.4**	**20.4**	92.7	28.8
FULL	33.3	27.4	20.9	105.3	36.6

Table 2: Average results for Recall@5 (**R**; Eqn. 1), METEOR (**M**; Denkowski and Lavie, 2014), SPICE (**S**; Anderson et al., 2016) , CIDEr (**C**; Vedantam et al., 2015), BLEU (**B**; Papineni et al., 2002). RR stands for re-ranking after decoding.

	Color		Size		Verb	
	A	I	A	I	T	I
SAT	3.7	10.5	0	0	1.6	2.2
BUTD	5.4	10.9	0.5	0	11.6	10.3
BUTR	6.4	16.2	0.3	0.2	7.0	8.6
+ RR	**13.8**	**26.0**	**1.4**	**0.8**	**20.3**	**16.9**
FULL	42.7	38.7	5.9	33.3	39.6	39.5

Table 3: Detailed Recall@5 scores for different categories of held out pairs. The scores are averaged over the set of scores for pairs from the respective category. RR stands for re-ranking after decoding. Color and size adjectives are split into **A**nimate or **I**nanimate objects; Verbs are split into **T**ransitive and **I**ntransitive verbs.

The Recall@5 scores for different categories of held out pairs is presented in in Table 3, and Figure 3 presents examples of images and the generated captions from different models. We observe that all models are generally best at describing colors, especially of inanimate objects; they nearly never correctly describe held out size modifiers; and for held out noun–verb pairs, performance is slightly better for transitive verbs.

Concepts	white horse	blue bus	small cat	small plane	man eat	bird stand
						
SAT	a black and white cow standing on top of a lush green field	a bus parked on the side of the street	a cat sitting on top of a wooden bench	a fighter jet on top of a lush green field	a man sitting at a table with a plate of food	a white bird sitting on top of a car
BUTD	a brown and white cow standing on a lush green field	a public transit bus on a city street	a cat sitting on top of a wooden bench	a white and green airplane on a field	a man sitting down with a plate of food	a white bird sitting on top of a car
BUTR+RR	a large **white horse** standing on top of a green field	a **blue** and yellow **bus** traveling down the street	a cat sitting on a bench near a wall	a white and green plane is parked on the grass	a **man** sitting down **eating** a plate of food	a large white **bird standing** on top of a car

Figure 3: Selected examples of the captions generated by SAT, BUTD, and BUTR for six different concept pairs. The **bold words** in a caption indicate compositional success.

7 Analysis and Discussion

Describing colors: The color–noun pairings studied in this work have the best generalization performance. We find that all models are better at generalizing to describing inanimate objects instead of animate objects, as shown in the detailed results in Table 3. One explanation for this could be that the colors of inanimate objects tend to have a higher variance in chromaticity when compared to the colors of animate objects (Rosenthal et al., 2018), making them easier to distinguish.

Describing sizes: The generalization performance for size modifiers is consistently low for all models. The CNN image encoders are generally able to predict the sizes of object bounding boxes in an image. However, this does not necessarily relate to the actual sizes of the objects, given that this depends on their distance from the camera. To support this claim, we perform a correlation analysis in Appendix F showing that the bounding box sizes of objects in the COCO dataset do not relate to the described sizes in the respective captions.

Nevertheless, size modification is challenging from a linguistic perspective because it requires reference to an object's comparison class (Cresswell, 1977; Bierwisch, 1989). A large mouse is so with respect to the class of mice, not with respect to the broader class of animals. To successfully learn size modification, a model needs to represent such comparison classes.

We hypothesize that recall is reasonable in the FULL setting because it exploits biases in the dataset, e.g. that trucks are often described as BIG. In that case, the model is not actually learning the meaning of BIG, but simple co-occurrence statistics for adjectives with nouns in the dataset.

Describing actions: In these experiments, the models were better at generalizing to transitive verbs than intransitive verbs. This may be because images depicting transitive events (e.g. eating) often contain additional arguments (e.g. cake); thus they offer richer contextual cues than images with intransitive events. The analysis in Appendix G provides some support for this hypothesis.

Diversity in Generated Captions: A crucial difference between human-written and model-generated captions is that the latter are less diverse (Devlin et al., 2015; Dai et al., 2017). Given that BUTR+RR improves compositional generalization, we explore whether the diversity of the captions is also improved. Van Miltenburg et al. (2018) proposes a suite of metrics to measure the diversity of the captions generated by a model. We apply these metrics to the captions generated by BUTR+RR and BUTD and compare the scores to the best models evaluated in Van Miltenburg et al. (2018).

The results are presented in Table 4. BUTR+RR shows the best performance as measured by most of the diversity metrics. BUTR+RR produces the highest percentage of novel captions (%Novel), which is important for compositional generalization. It generates sentences with a high average sentence length (ASL) – performing similarly to Liu et al. (2017) – but with a larger standard deviation, suggesting a greater variety in the captions. The total number of word types (Types) and cover-

Model	ASL	Types	TTR_1	TTR_2	%Novel	Cov	Loc_5
Liu et al. (2017)	**10.3** $\pm$ 1.32	598	0.17	0.38	50.1	0.05	0.70
Vinyals et al. (2017)	10.1 $\pm$ 1.28	953	0.21	0.43	90.5	0.07	0.69
Shetty et al. (2017)	9.4 $\pm$ 1.31	**2611**	0.24	0.54	80.5	**0.20**	0.71
BUTD	9.0 $\pm$ 1.01	1162	0.22	0.49	56.4	0.09	0.78
BUTR+RR	10.2 $\pm$ **1.76**	1882	**0.26**	**0.59**	**93.6**	0.14	**0.80**
Validation data	11.3 $\pm$ 2.61	9200	0.32	0.72	95.3	-	-

Table 4: Scores for diversity metrics as defined by Van Miltenburg et al. (2018) for different models.

age (Cov) are higher for Shetty et al. (2017), which is trained with a generative adversarial objective in order to generate more diverse captions. However, these types are more equally distributed in the captions generated by BUTR+RR, as shown by the higher mean segmented type-token ratio (TTR_1) and bigram type-token ratio (TTR_2).

The increased diversity of the captions may explain the lower BLEU score of BUTR+RR compared to BUTD. Recall that BLEU measures weighted n-gram precision, hence it awards less credit for captions that are lexically or syntactically different than the references. Thus, BLEU score may decrease if a model generates diverse captions. We note that METEOR, which incorporates non-lexical matching components in its scoring function, is higher for BUTR+RR than BUTD.

Decoding strategies: The failure of the captioning models to generalize can be partially ascribed to the effects of maximum likelihood decoding. Holtzman et al. (2019) find that maximum likelihood decoding leads to unnaturally flat and high per-token probability text. We find that even with grounding from the images, the captioning models do not assign a high probability to the sequences containing compositions that were not observed during training. BUTR is jointly trained with a ranking component, which is used to re-rank the generated captions, thereby ensuring that at the sentence-level, the captions are relevant for the image. It can thus be viewed as an improved decoding strategy such as those proposed in Vijayakumar et al. (2018); Fan et al. (2018); Radford et al. (2019); Holtzman et al. (2019).

8 Conclusion

Image captioning models are usually evaluated without explicitly considering their ability to generalize to unseen concepts. In this paper, we argued that models should be capable of *compositional generalization*, i.e. the ability to produce captions that include combinations of unseen concepts. We evaluated the ability of models to generalize to unseen adjective–noun and noun–verb pairs and found that two state-of-the-art models did not generalize in this evaluation, but that an image–sentence ranking model did. Given these findings, we presented a multi-task model that combines captioning and image–sentence ranking, and uses the ranking component to re-rank the captions generated by the captioning component. This model substantially improved generalization performance *without* sacrificing performance on established text-similarity metrics, while generating more diverse captions. We hope that this work will encourage researchers to design models that better reflect human-like language production.

Future work includes extending the evaluation to other concept pairs and other concept classes, analysing the circumstances in which the re-ranking step improves compositional generalization, exploring the utility of jointly trained discriminative re-rankers into other NLP tasks, developing models that generalize to size modifier adjectives, and devising approaches to improve the handling of semantically equivalent outputs for the proposed evaluation metric.

Acknowledgements

We thank Emiel van Miltenburg and Ákos Kádár for their extensive feedback on the work, and the reviewers, Ana Valeria Gonzales, Daniel Hershcovich, Heather Lent, and Mareike Hartmann for their comments. We also thank the participants of the Lorentz Center workshop on Compositionality in Brains and Machines for suggesting the phrase "paradigmatic gap". MN was supported by the Erasmus+ Traineeship program. RA and MA are funded by a Google Focused Research Award.

References

Harsh Agrawal, Karan Desai, Xinlei Chen, Rishabh Jain, Dhruv Batra, Devi Parikh, Stefan Lee, and Peter Anderson. 2018. nocaps: novel object captioning at scale. *arXiv preprint arXiv:1812.08658*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *EMNLP*, pages 936–945. Association for Computational Linguistics.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6077–6086.

Yuval Atzmon, Jonathan Berant, Vahid Kezami, Amir Globerson, and Gal Chechik. 2016. Learning to generalize to new compositions in image understanding. *CoRR*.

Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. 2018. Sod-mtgan: Small object detection via multi-task generative adversarial network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 206–221.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank. 2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *Journal of Artificial Intelligence Research*, 55:409–442.

Manfred Bierwisch. 1989. The semantics of gradation. In *Dimensional adjectives, ed. Manfred Bierwisch and Ewald Lang*, pages 71–261. Berlin: Springer-Verlag.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 793–802.

M. J. Cresswell. 1977. The semantics of degree. In *Montague grammar, ed. Barbara Partee*, pages 261–292. New York: Academic Press.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2970–2979.

Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J Gershman, and Noah D Goodman. 2018. Evaluating compositionality in sentence embeddings. *arXiv preprint arXiv:1802.04302*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*, pages 376–380.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 100–105.

Desmond Elliott and Frank Keller. 2014. Comparing automatic evaluation measures for image description. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 452–457.

Allyson Ettinger, Ahmed Elgohary, Colin Phillips, and Philip Resnik. 2018. Assessing composition in sentence vector representations. *arXiv preprint arXiv:1809.03992*.

Fartash Faghri, David J. Fleet, Jamie Kiros, and Sanja Fidler. 2018. VSE++: improving visual-semantic embeddings with hard negatives. In *British Machine Vision Conference 2018, BMVC 2018, Northumbria University, Newcastle, UK, September 3-6, 2018*, page 12.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. 2015. From captions to visual concepts and back. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1473–1482.

Jerry A Fodor and Ernest Lepore. 2002. *The compositionality papers*. Oxford University Press.

Jerry A Fodor and Zenon W Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. 2013. Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems*, pages 2121–2129.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Lisa Anne Hendricks, Subhashini Venugopalan, Marcus Rohrbach, Raymond Mooney, Kate Saenko, and Trevor Darrell. 2016. Deep compositional captioning: Describing novel object categories without paired training data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–10.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Keizo Kato, Yin Li, and Abhinav Gupta. 2018. Compositional learning for human object interaction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 234–251.

Mert Kilickaya, Aykut Erdem, Nazli Ikizler-Cinbis, and Erkut Erdem. 2017. Re-evaluating automatic metrics for image captioning. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 199–209.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Brenden M Lake and Marco Baroni. 2017. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv preprint arXiv:1711.00350*.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. 2017. Building machines that learn and think like people. *Behavioral and brain sciences*, 40.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Chang Liu, Fuchun Sun, Changhu Wang, Feng Wang, and Alan Yuille. 2017. Mat: A multimodal attentive translator for image captioning. *arXiv preprint arXiv:1702.05658*.

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2018. Neural baby talk. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7219–7228.

Junhua Mao, Xu Wei, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan L Yuille. 2015. Learning like a child: Fast novel visual concept learning from sentence descriptions of images. In *Proceedings of the IEEE international conference on computer vision*, pages 2533–2541.

Edward H Matthei. 1982. The acquisition of prenominal modifier sequences. *Cognition*, 11(3):301–332.

James L McClelland, David E Rumelhart, PDP Research Group, et al. 1986. Parallel distributed processing. *Explorations in the Microstructure of Cognition*, 2:216–271.

R Thomas McCoy, Tal Linzen, Ewan Dunbar, and Paul Smolensky. 2018. Rnns implicitly implement tensor product representations. *arXiv preprint arXiv:1812.08718*.

Ishan Misra, Abhinav Gupta, and Martial Hebert. 2017. From red wine to red tomato: Composition with context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1792–1801.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. *proceedings of ACL-08: HLT*, pages 236–244.

Richard Montague. 1974. Formal philosophy, ed. r. thomason. *New Haven*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Barbara Partee. 1984. Compositionality. *Varieties of formal semantics*, 3:281–311.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Steven Piantadosi and Richard Aslin. 2016. Compositional reasoning in early childhood. *PloS one*, 11(9):e0147734.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Isabelle Rosenthal, Sivalogeswaran Ratnasingam, Theodros Haile, Serena Eastman, Josh Fuller-Deets, and Bevil R Conway. 2018. Color statistics of objects, and color tuning of object cortex in macaque monkey. *Journal of vision*, 18(11):1–1.

Mohammad Amin Sadeghi and Ali Farhadi. 2011. Recognition using visual phrases. In *CVPR 2011*, pages 1745–1752. IEEE.

Rakshith Shetty, Marcus Rohrbach, Lisa Anne Hendricks, Mario Fritz, and Bernt Schiele. 2017. Speaking the same language: Matching machine to human captions by adversarial training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4135–4144.

Paul Smolensky. 1988. On the proper treatment of connectionism. *Behavioral and brain sciences*, 11(1):1–23.

Emiel Van Miltenburg, Desmond Elliott, and Piek Vossen. 2018. Measuring the diversity of automatic image descriptions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1730–1741.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575. IEEE Computer Society.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasaath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2017. Show and tell: Lessons learned from the 2015 mscoco image captioning challenge. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):652–663.

Liwei Wang, Alexander Schwing, and Svetlana Lazebnik. 2017. Diverse and accurate image description using a variational auto-encoder with an additive gaussian encoding space. In *Advances in Neural Information Processing Systems*, pages 5756–5766.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France. PMLR.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659.

Representing Movie Characters in Dialogues

Mahmoud Azab[1], **Noriyuki Kojima**[1], **Jia Deng**[2], **Rada Mihalcea**[1]

[1]Computer Science and Engineering
University of Michigan, Ann Arbor
{mazab,kojimano,mihalcea}@umich.edu

[2]Department of Computer Science
Princeton University
jiadeng@princeton.edu

Abstract

We introduce a new embedding model to represent movie characters and their interactions in a dialogue by encoding in the same representation the language used by these characters as well as information about the other participants in the dialogue. We evaluate the performance of these new character embeddings on two tasks: (1) character relatedness, using a dataset we introduce consisting of a dense character interaction matrix for 4,761 unique character pairs over 22 hours of dialogue from eighteen movies; and (2) character relation classification, for fine- and coarse-grained relations, as well as sentiment relations. Our experiments show that our model significantly outperforms the traditional Word2Vec continuous bag-of-words and skip-gram models, demonstrating the effectiveness of the character embeddings we introduce. We further show how these embeddings can be used in conjunction with a visual question answering system to improve over previous results.

1 Introduction

Understanding characters (or more broadly people) plays a critical role in the human-level interpretation of dialogues – be those in stories, movies, or day-to-day conversations. The verbal interaction between characters provides important information (Iyyer et al., 2016; Elson et al., 2010). In these contexts, the names of characters trigger reasoning at a much deeper level than other regular words, due to the character background, behaviors, social network, and so forth. Currently, the most commonly used word embedding models such as Word2Vec (Mikolov et al., 2013a,b) and Glove (Pennington et al., 2014) represent characters using the embeddings corresponding to the tokens used to name them. Using these models in a dialogue setting to represent the characters poses three main issues. First, name mentions in dialogues are sparse (Azab et al., 2018), which makes it difficult for these models to learn a good quality representation for these names (Barteld, 2017). Second, in dialogues or narratives, names often do not refer to the same person, and yet these embeddings have a single vector representation for each word in the vocabulary. For example, "Danny" in the dialogue of the "American History X" movie is different from "Danny" in the "Ocean's Eleven" movie. Finally, the learned embeddings of these names reflect the co-occurrences of these name mentions and other words uttered by these characters, but do not model how related these characters are. Thus, the resulting embeddings cannot be effectively used to further reason about the characters and their relations.

Henry:	I did not know you could fly a plane.
Indiana:	Fly yes. Land no. Dad, you have to use the machine gun. Get it ready. Eleven o'clock!
Henry:	What happens at eleven o'clock?
Indiana:	Twelve, eleven, ten. Eleven o'clock, fire! Dad, are we hit?
Henry:	More or less. Son, I am sorry. They got us.
Indiana:	Hang on, dad. We are going in.

Table 1: A snippet of conversation between two characters from the "Indiana Jones and the Last Crusade" movie with each dialogue turn annotated with its corresponding speaker name. We aim to generate embedding representations for "Indiana" and "Henry" in a way that captures their relation.

The representation of characters in dialogues has been an important task for social network extraction (Elson et al., 2010), character relation modeling (Chaturvedi et al., 2016), and persona-based conversation models (Li et al., 2016). However, most of the previous work relies upon the ex-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 99–109
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

traction of linguistic features like explicit forms of address (Makazhanov et al., 2014), the length of the utterance, or the frequency of exchanges between the characters (Elson et al., 2010).

In this work, we address the task of representing characters in dialogues, specifically focusing on movies and plays. Given a set of dialogue turns, annotated with the corresponding speaker names, our goal is to generate a vector representation for each of these characters that captures the relation with other characters. We propose a new approach to embed characters in dialogues based not only on what a character is saying, but also to whom. This model allows the information from the words in a dialogue turn to propagate to the representation of the previous and following speakers.

Despite its simplicity, our model yields strong empirical performance. By evaluating our model on two different tasks – namely character relatedness and character relation classification (fine-grained, coarse-grained, and sentiment) – we find that the model exceeds by a large margin several strong baselines, which indicates that our model effectively captures the various characteristics of characters. Additionally, in the process of evaluating the model, we build a new dataset consisting of 4,761 character relation pairs obtained from eighteen movies, manually annotated with relatedness scores and relations of various granularities. We are making the dataset publicly available.

2 Related Work

Learning distributional representation of words plays an increasingly important role in representing text in many tasks (Bengio et al., 2013; Chen and Manning, 2014). The existence of huge datasets allowed learning high quality word embeddings in an unsupervised way by training a neural network on fake objectives (Mikolov et al., 2013a,b; Turney and Pantel, 2010). A major strength of these learned word embeddings is that they are able to capture useful semantic information that can be easily used in other tasks of interest such as semantic similarity and relatedness between pair of words (Mikolov et al., 2013a; Pennington et al., 2014; Wilson and Mihalcea, 2017) and dependency parsing (Chen and Manning, 2014; Dyer et al., 2015). However, these models treat names and entities no more than the tokens used to mention them. As a result, these models are unable to well represent names in narrative understanding task because the word "John" in a given story can be very different from the word "John" in another narrative. In this work, we only focus on representing character names and not the whole embedding space (Ji et al., 2017).

Recently, several approaches have been proposed to build dynamic representations for entities (Henaff et al., 2016; Ji et al., 2017; Kobayashi et al., 2016, 2017). One common approach is to rely on neural language models to encode the local context of an entity and use the resulting context vectors as the embedding for subsequent occurrences of that entity (Kobayashi et al., 2016, 2017). Another approach is to learn a generative model that generates the representation of an entity mention (Ji et al., 2017). Henaff et al. (2016) proposed an explicit entity tracking model by relying on an external memory to store information about entities as they appear in a given sentence. While these rich representations improve the performance on several tasks such as coreference and reading comprehension, they rely on explicit mentions of entities in text as available in toy datasets such as bAbi (Weston et al., 2015). Thus, it is difficult to apply these representations in a dialogue setting due to the sparseness of name mentions in dialogue, as well as the lack of explicit conversation connections between characters (as available in movies) (Azab et al., 2018). Most of the existing story understanding work feeds the model with the vector representations of names based on a global model such as Word2Vec or Glove, which hinders the ability of these models to understand dialogue (Tapaswi et al., 2016; Na et al., 2017; Lei et al., 2018). Recently, Li et al. (2016) relied on TV series scripts in order to learn speaker persona representations and used these representations to improve the performance of neural conversation models. Unlike (Ji et al., 2017; Li et al., 2016), we focus on representing character names in dialogue settings and learning different embeddings for characters from different story dialogues in a way that reflects the relatedness of story characters; more specifically, we propose the use of speaker prediction as an auxiliary supervision to improve the character representation.

Identifying and analyzing character relations in literary texts is a well studied problem (Agarwal et al., 2013; Makazhanov et al., 2014; Elson et al., 2010; Iyyer et al., 2016). Most of these models depend on analyzing the co-occurrence of the char-

acters and stylistic features used while characters address each other. These models are really important to summarize, understand, and generate stories (Elson et al., 2010). In this work, we use the task of character relation classification as an extrinsic evaluation task to evaluate the impact of character embeddings on this task.

3 Character Embeddings

Characters play an important role in any dialogue, including movies or plays. Yet, work to date has rarely considered specialized character representations. We hypothesize that a representation that leverages both the language uttered by the characters as well as information on the other characters in the dialogue could result in richer encodings. The intuition behind our hypothesis is explained by the example in table 1. Here, the word "Dad" should be associated not only with "Indiana" but also propagate its information to "Henry", conditioned by "Indiana". Our proposed model is well conveying this intuition to encode characters.

3.1 Setup

Our architecture builds on a pretrained embedding model generated by standard Word2Vec models (Mikolov et al., 2013a,b) or pre-trained contextualized word representations from neural language models (ELMo) (Peters et al., 2018). We start by collecting sets of (current speaker, previous speakers, next speakers, context words) as training examples. We split the four elements in the sets into target and context depending on our objectives. Figure 1 describes the input-output (target-context) pairs of our system. Additionally, our model works as an unsupervised post-training of existing embeddings, rather than starting the training from scratch. This is due to the fact that getting a good representation for characters is a separate task from getting a general representation of tokens. A good pre-trained embedding space is an essential component to map characters so that they will be distributed in a semantically meaningful embedding space. While a good pre-trained embedding is important, our models focus on "moving" the character embeddings without affecting any other word representations.

3.2 Architecture

We propose two post-training schemes, which we refer to as Character Embedding (SG) and Character Embedding (CBOW). The differences stand in the objective of post-training, given sets of (current speaker, previous speakers, next speakers, context words) as training examples. Formally, given the sequence of speakers at each turn $S = s_1, s_2, s_3, , , s_{T-1}, s_T$, we define context words C for turn t as the set of words found by a sliding context window in the utterance. We propose our post-training objectives as following:

$$L = \frac{1}{N} \sum_{s_i \in S} \sum_{w_i \in C(s_i)} \sum_{-sw \leq j \leq sw} \log(p(w_i|s_{i+j}))) \quad (1)$$

$$L = \frac{1}{N} \sum_{s_i \in S} \sum_{w_i \in C(s_i)} (\log(p(s_i|w_i) + \sum_{-sw \leq j \leq sw, j \neq 0} \log(p(s_i|s_{i+j}))) \quad (2)$$

Our Character Embedding (SG) model maximizes the objective on Equation 1, while Character Embedding (CBOW) maximizes the objective on Equation 2, where N indicates the number of training examples and sw indicates the size of the speaker window (speaker window of size one means we consider speakers of one preceding turn and one succeeding turn). Our formulation defines probabilities $p(s_i|w_i)$, $p(s_i|s_{i+j})$ and $p(w_i|s_{i+j})$ using the softmax equation. We also define two transformations of our network – lookup table (LUT) initialized by embedding of pre-trained embedding model and Linear Projection Layer W.

To examine the generality of our post-training schemes, we also apply them to another pre-trained word embedding model. Given a dialogue turn, we encode it using ELMo's pre-trained Bi-LSTM model (Peters et al., 2018) to generate a sequence of contextualized vectors for words. We add a linear projection layer on top that takes the generated embedding, in addition to the previous and following speakers, and train it to predict the speaker of the current turn. We refer to this model as Character Embedding (ELMo).

3.3 Training

We represent our contexts and targets as a one hot vector of length equal to the vocabulary size. The purpose of our model is to update the embedding

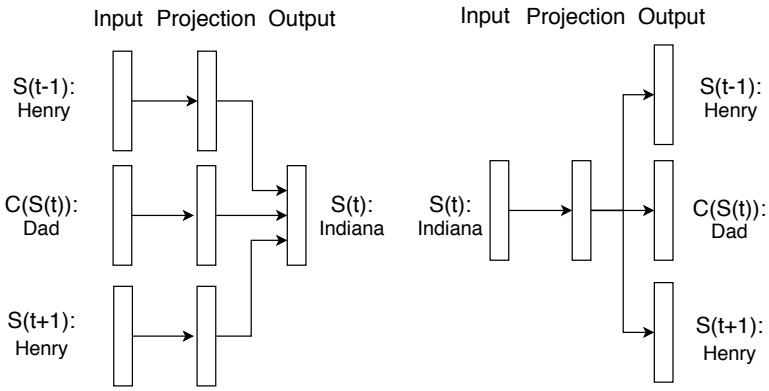


Figure 1: The conceptual figure describing input /output pairs of our character embedding model. The diagram describes when both the speaker window and the context window are size one. **Left**: Character Embedding(CBOW), **Right**: Character Embedding(SG).

of characters in LUT by propagating the gradient from our objectives. We use cross-entropy to calculate the loss, and we use gradient descent to update the parameters. The description of our Character Embedding (SG) model with a speaker window size of one is showed in Algorithm 1.

4 Evaluation Tasks and Datasets

We evaluate the quality of our speaker embedding model across two different tasks. Our goal is to evaluate how well each embedding model captures simple and complex character representations and interactions.

4.1 Character Relatedness

Measures of semantic relatedness between words indicate the degree to which words are associated with any kind of semantic relationship such as synonymy, antonymy, and so on. Semantic relatedness is commonly used as an absolute intrinsic evaluation task to assess and compare the quality of different word embeddings (Schnabel et al., 2015; Yih and Qazvinian, 2012; Upadhyay et al., 2016) and phrase embeddings (Wilson and Mihalcea, 2017).

Similarly, we define character relatedness as the degree to which a pair of characters in a given story are related to each other based on the story plot and their level of interaction throughout the dialogue. Given a pair of characters, we would like the relatedness score between their embedding representations to have a high correlation with their corresponding human-based relatedness score. Thus, the distance of the embeddings between closely related characters should be smaller than the distance between less related ones.

To measure the relatedness between characters in movies, we construct a new annotated dataset based on a publicly available dataset (Azab et al., 2018). That dataset includes 28K turns spoken by 396 different speakers in eighteen movies covering different genres, with the subtitles of each movie labeled with the character name of their corresponding speakers. On average, each character uttered 452 words.

For each movie in that dataset, two human annotators watched the movies and annotated a dense relatedness matrix of characters on a 1-5 scale. Table 2 shows the meaning of each score. These scores reflect the level of interaction or how closely related the characters are over the course of the movie. For example, given two characters X and Y, a high score for X and Y is assigned if e.g., X is the father of Y, regardless of the amount of interaction between the two characters. We also give a high score for the cases where X and Y are closely interacted, even if they are unrelated in terms of kinship. Due to the sparseness of the number of closely related characters, we asked the annotators to select the higher score when hesitating between two scores.

For three movies, the Pearson correlation between the two annotators is 0.8394, which reflects a very good agreement. We then average the scores assigned by the annotators and use the result as the human relatedness ground-truth score for each pair of characters.

In this dataset, we have 4,761 unique character pairs annotated with a relatedness score. Figure 2 shows the statistics over the relatedness scores. As shown in the table, only a small number of character pairs are closely related, while the majority

Algorithm 1: Character Embedding(SG)

E: The embedding from pre-trained model
W: Linear Projection Layer
α: Learning Rate
maxepoch: maximum epoch to run
LUT $\leftarrow$ E, epoch $\leftarrow$ 1;
while $epoch \leq maxepoch$ **do**
  for *t from 2 to* $T-1$ **do**
    $x_1 \leftarrow LUT[s_{t-1}]$;
    $x_2 \leftarrow LUT[s_t]$;
    $x_3 \leftarrow LUT[s_{t+1}]$;
    for w_0 *in* $C(s_t)$ **do**
      target $\leftarrow LUT[w_0]$;
      logits $= tanh(W^T(x_1+x_2+x_3))$;
      prediction $= softmax$(logits);
      loss $= -$target $+ \log$(prediction);
      $W := W - \alpha * \frac{\delta loss}{\delta W}$;
      $LUT[s_{t-1}] := x_1 - \alpha * \frac{\delta loss}{\delta x_1}$;
      $LUT[s_t] := x_2 - \alpha * \frac{\delta loss}{\delta x_2}$;
      $LUT[s_{t+1}] := x_3 - \alpha * \frac{\delta loss}{\delta x_3}$;
    end
  end
  epoch := epoch + 1
end

5	interacted frequently/closely related
4	interacted/related
3	moderately interacted/somewhat related
2	interacted few times/not related
1	did not interact/not related

Table 2: Relatedness annotation scores.

of the characters have either interacted very few times or did not interact at all. However, it is important to include these unrelated pairs while evaluating the quality of the character embeddings, as unrelated pairs might be closer than related ones especially for minor characters that do not speak much during the dialogue.

4.2 Character Relationships

Understanding the relationships between characters is a primary task in extracting and analyzing social relation networks from literary novels (Elson et al., 2010; Agarwal et al., 2013). It is also important for improving computational story summarization and generation methods (Elsner, 2012; Gorinski and Lapata, 2015).

Character relationship is a more complex task than character relatedness. In this task, given a pair of character embeddings, we would like to classify the type of their relationship on multiple dimensions. Specifically, we consider: fine-grained relations, such as sister/father/friend/enemy; coarse-grained relations, such as familial/social/professional; and relation sentiment, i.e., positive, negative or neural. The goal of this task is to evaluate the quality of our character embeddings and how well it captures such complex information in an unsupervised fashion. It also serves as an extrinsic evaluation for the impact of our character representations on downstream tasks.

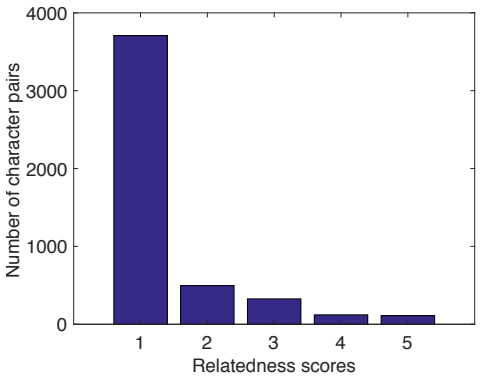


Figure 2: Statistics of the character relatedness dataset on movies of speaker naming dataset.

We use a subset of character relationships in a literary dataset (Massey et al., 2015). This dataset includes annotations for eighteen fine-grained relationship classes, four coarse-grained relationship classes, and three relation sentiment classes.[1] We use the 31 Shakespeare plays in this dataset, and obtain their corresponding text from project Gutenberg. We use the Shakespeare plays because they have the dialogue turns annotated with speakers names, which is necessary for training our character embedding models. The plays include a total of 605 character pair relationship annotations.

5 Experiments

5.1 Baselines

For each task, we compare our character embedding models against five baselines:

[1]Annotations on temporal change in the sentiment between each pair of characters is also included, but since our models do not have the ability to track such temporal information, we do not use these annotations.

Interaction Frequency. We count the number of exchanged dialogue turns between every pair of characters and normalize it by the total number of turns spoken by a given pair of characters.

TF-IDF. We treat all the utterances of a character as a document and calculate a tf-idf weight for each word. We then represent a character by its tf-idf vector of the words that they uttered.

Word2Vec (CBOW) model. We use the traditional Word2Vec architecture to train a word embedding space based on the continuous bag-of-words approach (Mikolov et al., 2013a). Given a sequence of words D, the context words that exist in a defined window size are considered as input to the network and the objective is to predict the target word by maximizing the average long probability:

$$L = \frac{1}{|D|} \sum_{w_i \in D} \log P(w_i | C(w_i)) \quad (3)$$

Word2Vec (SG) model. We use the skip-gram architecture of Word2Vec with negative sampling (Mikolov et al., 2013b). In this architecture, the objective is to learn a representation of the target word that would be good at predicting the words within a defined window by maximizing the average log probability:

$$L = \frac{1}{|D|} \sum_{w_i \in D} \sum_{w_0 \in C(w_i)} \log P(w_0 | w_i) \quad (4)$$

Character BOW. We represent each character as the mean-pooling of a 300-dimension pre-trained Word2Vec representation of all the words that this character has uttered through the entire dialogue.

Doc2Vec. We train a Doc2Vec model (Le and Mikolov, 2014) as tagged documents using the character names as the document tags. We then represent each character as the Doc2Vec representation of all the words that this character has uttered through the entire dialogue.

ELMo (Mean-Pooling). We use pre-trained contextualized word representations from neural language models (ELMo) (Peters et al., 2018) to generate character names representations based on the sentences that include their names.[2] To generate these representations, we feed the pre-trained ELMo model with a Glove representation for the words and ELMo augments their representation with the hidden states of its two layers bi-directional LSTM to represent the words with respect to their context. For each character name, we average their contextualized representations through the entire dialogue.

[2]We also tried training ELMo from scratch on our data but the pre-trained model produces better results.

5.2 Experimental Setting

To have these models trained on in-domain data, we use GenSim (Řehůřek and Sojka, 2010) to train the different architectures of Word2Vec on the almost 600K sentences / 4M words of subtitles and Shakespeare plays. For the target movies and plays, the speaker names are included in the training data so that we can have a vector representation for each character name. The names in our corpus have been manually normalized so that 'Joe' and 'Joseph' in a movie get the same representation, while 'Joseph' in a different movie gets a different representation. To achieve the first part of the name normalization, we utilize the name-clustering algorithm provided by Bamman (2014) to extract and cluster name tokens from the text and annotate the true representation of names for each cluster. We achieve the second part of the name normalization by adding the text title to the name tokens (e.g., 'Michael' becomes 'Michael$_{Othello}$').

For GenSim (Řehůřek and Sojka, 2010), we set the learning rate to 0.1, the window size to 4 and the samples to 50 for negative sampling. We run 30 epochs to train our baselines. For post-training by our models, we use a gradient decent to update our parameters. For general experiments, we set the learning rate to 0.1 and the learning rate decays by the factor of 0.9 per 10 epochs. We run maximum 40 epochs for our post-training. For Character Embedding (CBOW), we use a context window of size two. We use a speaker window of size one for both the Character Embedding (CBOW) and the Character Embedding (SG).

5.3 Results

Character Relatedness. For each model, given a pair of characters we compute the cosine similarity score between the embeddings of these two characters, defined as:

$$similarity(\boldsymbol{C}1, \boldsymbol{C}2) = \frac{\boldsymbol{C}1 \cdot \boldsymbol{C}2}{||\boldsymbol{C}1|| \cdot ||\boldsymbol{C}2||} \quad (5)$$

and compute the similarity score between two characters in the embedding space similar to (Col-

Movie	Character	Methods	Closest	Second closest	Third Closest
The Devil's Advocate	Alice Lomax	Ground Truth	**Kevin Lomax**	**John Milton**	**Mary Lomax**
		Interaction Frequency	**Kevin Lomax**	Pam Garrety	John Milton
		TF-IDF	Mary Lomax	John Milton	Don King
		Character Average BOW	John Milton	Kevin Lomax	Barbara
		Word2Vec (CBOW)	Lloyd Gettys	Judge Poe	Alexander Cullen
		Word2Vec (SG)	Alfonse D'amato	Lloyd Gettys	Judge Poe
		ELMo (Mean-Pooling)	**Kevin Lomax**	Mary Lomax	Alexander Cullen
		Character Embedding(CBOW)	**Kevin Lomax**	Judge Poe	**Mary Lomax**
		Character Embedding(SG)	**Kevin Lomax**	**John Milton**	**Mary Lomax**
		Character Embedding(ELMo)	**Kevin Lomax**	Pam Garrety	**Mary Lomax**

Table 3: Example of character relatedness task. Given a character, we list the top three characters sorted in descending order from left to right according to their similarity scores.

lobert et al., 2011; Mikolov et al., 2013b). The list of the nearest characters of a given character C are all the other characters from the same movie sorted in descending order by their similarity score with respect to C.

	Pearson Coeff
Interaction Frequency	0.3632
TF-IDF	0.3129
Doc2Vec	0.1771
Word2Vec (CBOW)	0.2081
Word2Vec (SG)	0.1989
Character BOW	0.2256
ELMo (Mean-Pooling)	0.3212
Character Embedding(CBOW)	**0.4644**
Character Embedding(SG)	**0.4933**
Character Embedding(ELMo)	**0.3475**

Table 4: Comparison between the average Pearson correlation coefficient scores of the different models against average human relatedness scores.

Table 4 shows the Pearson correlation coefficients of the resulting similarity scores of each model against the average human annotation scores. These results suggest that having the context window over the utterance and adding the previous and next speakers to the input layer greatly improves the ability of the character embeddings to capture the relatedness between the different characters in a given story dialogue.

Table 3 shows an example of characters that are most related to "Alice Lomax" from the movie "The Devil's Advocate" as calculated based on each model sorted in descending order according to their cosine similarity scores. It is worth noting that Kevin Lomax is Alice's son, John Milton is Kevin's father and Mary Ann Lomax is Kevin's wife. On the other hand the characters suggested by both Word2Vec CBOW and SG models did not interact with Alice through the whole movie.

To further analyze the quality of the produced character embeddings, we evaluate the embeddings across different characters according to the their frequency of appearance in the movies. Figure 3 shows a comparison between the performance of the different models over minor and major characters based on the number of dialogue turns that each character uttered. These results show that our character embedding model consistently outperforms the traditional Word2Vec baseline models and reflect the robustness of our model in generating better character embeddings.

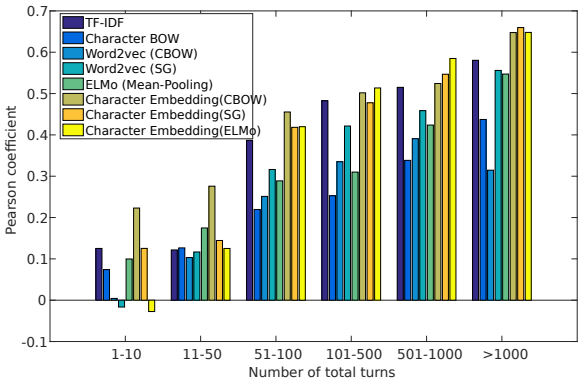


Figure 3: Comparison of the average Pearson correlation coefficient over characters who had different number of turns.

Character Relationship. We have three classification tasks for character relationships: 1) fine-grained relationship classification; 2) coarse-grained relationship classification; 3) relation sentiment classification. For each of these tasks, we train a logistic regression classifier using the Scikit-learn library (Pedregosa et al., 2011). These classifiers take a pair of character embeddings as a concatenation of their vectors and predict their

	Fine-grained Relation			Coarse-grained Relation			Sentiment		
	P	R	F	P	R	F	P	R	F
Interaction Frequency	0.04	0.16	0.06	0.30	0.44	0.33	0.33	0.58	0.42
TF-IDF	0.11	0.12	0.10	0.39	0.42	0.40	0.43	0.53	0.40
Character Average BOW	0.08	0.16	0.05	0.33	0.43	0.28	0.28	0.53	0.37
Word2Vec (CBOW)	0.11	0.13	0.12	0.37	0.38	0.38	0.39	0.40	0.39
Word2Vec (SG)	0.09	0.12	0.10	0.37	0.37	0.37	0.41	0.43	0.42
Doc2Vec	0.12	0.12	0.12	0.40	0.40	0.40	0.42	0.42	0.42
ELMo (Mean-Pooling)	0.14	0.18	0.14	0.39	0.41	0.40	0.44	0.50	0.46
Character Embedding(CBOW)	0.11	0.14	0.12	0.43	0.44	0.43	0.44	0.47	0.44
Character Embedding(SG)	0.11	0.17	0.12	0.43	0.46	0.42	0.40	0.51	0.42
Character Embedding (ELMo)	**0.18**	**0.19**	**0.19**	**0.48**	**0.48**	**0.48**	**0.48**	**0.48**	**0.48**

Table 5: Comparison between the average of the precision, recall and macro-weighted f-score of the baselines and our character embedding model on both fine-grained, coarse-grained character relation and sentiment classification.

Play	Char 1	. Char 2	Methods	Fine-grained	Coarse-grained	Senti-ment
			Ground Truth	**lovers**	**social**	**positive**
			Interaction Frequency	**lovers**	**social**	**positive**
The Two			TF-IDF	servant	**social**	negative
Gentlemen	Julia	Proteus	Character Average BOW	friend	**social**	**positive**
of Verona			Word2Vec (CBOW)	servant	familial	negative
			Word2Vec (SG)	servant	familial	**positive**
			ELMo (Mean-Pooling)	friend	**social**	**positive**
			Character Embedding(CBOW)	**lovers**	**social**	negative
			Character Embedding(SG)	**lovers**	**social**	**positive**
			Character Embedding(ELMo)	**lovers**	**social**	**positive**

Table 6: Example of classification task on Shakespeare's play, using different baselines and our character representation methods. The classification output consists of the relations of character 2 from character 1's perspective. A bold face indicates a correct relation classification.

relationship. We use a leave-one-play-out cross-validation in which character pairs from each play are used as a test set and character pairs from the other plays are used to train the models. Table 5 shows the classification average precision, recall and weighted F-score obtained by training the logistic regression classifiers using the character embeddings produced by the different models. Training classifiers using our character embedding models consistently outperforms the classifiers trained using the other models, which reflects the quality of the semantic information captured by our character embeddings when compared to other models. Table 6 shows examples of the three character relation classification tasks as classified by our character embedding models and the baselines.

Question Answering. As a final evaluation, we test the impact of our character embedding on dialogue understanding. TVQA (Lei et al., 2018) is a challenging dataset that includes 152.5K multiple choice question answers about 21.8K video clips from 6 TV shows such as the *Big Bang Theory*, *House*, and so on. These questions were created in a way that requires understanding of both the dialogue and the visual content of a given video. Each video clip includes the video frames and subtitles with speaker names aligned automatically with their corresponding show scripts (around 69% of the subtitle segments include speakers names). We follow the same dataset splits for training, validation, and test.

	Accuracy	
	Q+S	Q+S+V
MS (Glove) (Lei et al., 2018)	0.6515	0.6770
MS (Glove w/o names)	0.6177	0.6467
MS (CharEmbedding(CBOW))	**0.6590**	**0.6852**
MS (CharEmbedding(SG))	**0.6554**	**0.6884**

Table 7: Comparison on the TVQA validation dataset using the MS method with Glove and Glove fine-tuned using our proposed character embedding method.

To evaluate our embedding, we use the baseline implementation proposed with the TVQA dataset, namely Multi-Stream (MS). This model relies on bidirectional attention between context (represented by subtitles and/or visual content) and question answer pairs as queries to predict the correct answer (Lei et al., 2018). Visual features are included as textual labels of detected visual concepts in the frames of the video clip. To measure the effect of the person names on the model, we apply a named entity recognizer and replace the names with a fixed randomly generated embedding. Table 7 shows the results from the MS method using Glove, Glove with removing names from subtitles, and using a fine-tuned Glove using our character embedding model. The use of our character embeddings bring improvements over the pre-trained Glove embeddings, which demonstrates the usefulness of these character representations.

6 Conclusion

In this paper, we presented a novel unsupervised embedding model to represent characters and their interaction in a dialogue. Our embedding model produces character representations that reflect the language used by the characters as well as information about their relations with other characters. To evaluate the performance of our character embeddings, we experimented with two tasks on two datasets: (1) character relatedness, using a dataset we introduced consisting of a dense character interaction matrix for 4,761 unique character pairs over 22 hours of dialogue extracted from 18 movies; and (2) character relation classification, for fine- and coarse-grained relations, as well as relation sentiment. Our experiments show that our model significantly outperforms the traditional Word2Vec continuous bag-of-words and skip-gram models, thus demonstrating the effectiveness of the character embeddings we introduced. We further showed how the character embeddings can be used in conjunction with a visual question answering system to improve over previous results.

The dataset annotated with character relatedness scores introduced in the paper is publicly available from http://lit.eecs.umich.edu/downloads.html.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by a Samsung research grant and by a DARPA grant HR001117S0026-AIDA-FP-045.

References

Apoorv Agarwal, Anup Kotalwar, and Owen Rambow. 2013. Automatic extraction of social networks from literary text: A case study on alice in wonderland. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

Mahmoud Azab, Mingzhe Wang, Max Smith, Noriyuku Kojima, Jia Deng, and Rada Mihalcea. 2018. Speaker naming in movies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

David Bamman. 2014. book-nlp: Natural language processing pipeline that scales to book-length documents. https://github.com/dbamman/book-nlp.

Fabian Barteld. 2017. Detecting spelling variants in non-standard texts. In *Proceedings of the Student Research Workshop at the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*.

Snigdha Chaturvedi, Shashank Srivastava, Hal Daumé III, and Chris Dyer. 2016. Modeling evolving relationships between characters in literary novels. In *AAAI*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Micha Elsner. 2012. Character-based kernels for novelistic plot structure. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

David K Elson, Nicholas Dames, and Kathleen R McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics.

Philip John Gorinski and Mirella Lapata. 2015. Movie script summarization as graph-based scene extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. *arXiv preprint arXiv:1708.00781*.

Sosuke Kobayashi, Naoaki Okazaki, and Kentaro Inui. 2017. A neural language model for dynamically representing the meanings of unknown words and entities in a discourse. *arXiv preprint arXiv:1709.01679*.

Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Dynamic entity representation with max-pooling improves machine reading. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. 2018. Tvqa: Localized, compositional video question answering. *arXiv preprint arXiv:1809.01696*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

Aibek Makazhanov, Denilson Barbosa, and Grzegorz Kondrak. 2014. Extracting family relationship networks from novels. *arXiv preprint arXiv:1405.0603*.

Philip Massey, Patrick Xia, David Bamman, and Noah A Smith. 2015. Annotating character relationships in literary texts. *arXiv preprint arXiv:1512.00728*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.

Seil Na, Sangho Lee, Jisung Kim, and Gunhee Kim. 2017. A read-write memory network for movie story understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. *arXiv preprint arXiv:1604.00425*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Steven Wilson and Rada Mihalcea. 2017. Measuring semantic relations between human activities. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*.

Wen-tau Yih and Vahed Qazvinian. 2012. Measuring word relatedness using heterogeneous vector space models. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Cross-lingual word embeddings and the structure of the human bilingual lexicon

Paola Merlo
University of Geneva
Paola.Merlo@unige.ch

Maria A. Rodriguez
University of Geneva
Maria.AnduezaRodriguez@unige.ch

Abstract

Research on the bilingual lexicon has uncovered fascinating interactions between the lexicons of the native language and of the second language in bilingual speakers. In particular, it has been found that the lexicon of the underlying native language affects the organisation of the second language. In the spirit of interpreting current distributed representations, this paper investigates two models of cross-lingual word embeddings, comparing them to the shared-translation effect and the cross-lingual coactivation effects of false and true friends (cognates) found in humans. We find that the similarity structure of the cross-lingual word embeddings space yields the same effects as the human bilingual lexicon.

1 Introduction

Research on the bilingual lexicon has uncovered fascinating interactions between the L1 (native language) and L2 (second language) lexicons showing that both production and comprehension coactivate lexical items in both languages, indicating that bilinguals store lexical representations from their native and their second language in the same space (Kroll and Dijkstra, 2012; Williams, 2014).[1]

This paper presents the first bilingual investigation of models of cross-lingual word embeddings and asks whether the bilingual spaces they define have similar properties to the human bilingual lexicon. Among the many questions and results in the vast bilingualism literature, we concentrate on coactivation effects in items with shared translations. We also study interference or facilitatory effects in form-meaning mapping, the case of *false friends*, words that share form but differ in meaning across the two languages, and *true friends*, words that share both form and meaning. We find that the similarity structure of cross-lingual word embeddings matches well with known experimental findings of the human bilingual lexicon.

2 The structure of the bilingual lexicon

The core findings about the bilingual lexicon confirm that the two languages occupy an integrated space and they interact with each other (Wolter, 2001). For example, both in the monolingual and bilingual lexicon the best predictor of the time to recognise a word is the number of similarly spelled words, within and across languages (Johnson and Pugh, 1994; van Heuven et al., 1998).[2] This implies that, functionally, the bilingual lexicon is an integrated system. Specifically, it has been proposed that languages do not simply share graphemes or phonemes, but that the lexicon, monolingual, bilingual or multi-lingual, is a space of distributed word representations where word forms from different languages map onto a common abstract conceptual code (Van Hell and de Groot, 1998). This general structural and functional assumption explains many findings. We concentrate here on two sets of coactivation effects.

[1] Throughout this paper, and following the current literature on the topic, we use the term 'bilingual' loosely, to refer to any speaker of more than one language. Although there has been much research on all aspects of bilingualism, and at all stages of proficiency, the effects we model here have been found in experiments testing speakers who began to learn their second language after their first, usually in a school context, and who are at an advanced level of proficiency. The form German-English below will indicate, for example, a native speaker of German who learnt English as a second language (Williams, 2014).

[2] Monolingual work has provided a finer-grained picture of this result, modulated by number of word senses and the semantic closeness of sense extensions, but the main result remains valid (Rodd et al., 2002).

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 110–120
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

2.1 The shared translation effect

Polysemous words that have many translations, such as English *bank* translated as *banca* (financial institution) or *riva* (river bank) in Italian, coactivate the correspondences for all the word senses, with various effects. One-to-many translations have been shown to slow down acquisition and processing for Italian-English bilinguals (Degani and Tokowicz, 2010), and to slow down response times of German-English speakers in anomaly detection tasks (Elston-Güttler and Williams, 2008).[3]

We are mainly interested in the result of Degani et al. (2011), as it concerns similarity spaces in shared translations. Degani et al. (2011) asked Hebrew-English bilinguals to rate the semantic relatedness of English word pairs that shared a translation in Hebrew (e.g., *tool* and *dish* both translated into Hebrew *kli*). Compared to both English pairs with different Hebrew translations, and to ratings by monolingual English speakers, bilinguals judged shared-translation pairs as more related in meaning (the shared-translation effect).[4]

2.2 Form-meaning mappings in translation

Competition (and facilitation) effects have been found both in comprehension and production depending on convergence and divergence of form-meaning mappings in translation. Recall that *false friends* are cross-linguistically similar in form but not in meaning, such as the English-Italian *estate*, which in Italian means *summer*, and *true friends* are words that share both (orthographic or phonological) form and meaning, such as English-French *glucose* or *danger*, in translation.

False friends effect In a cross-modal picture decision task, Weber and Cutler (2004) find that Dutch-English speakers are slower in matching an English word *(desk)* with the corresponding picture if the target picture's word form matches the Dutch form of one of the alternative pictures *(deskel = lid)*. It should be noted, however, that while the decision time was slower, the decision accuracy was not. Bilingual speakers do know which is the right word-picture match and perform accurately. Also, for English-Dutch false friends like *rust* (*rest* in Dutch) lexical decision times are slower than expected, if the list in which they are embedded also contains words from the other language (Dijkstra et al., 1998; Smits et al., 2006).

True friends effects In recognition, Dutch-English bilinguals performing a lexical decision task in English were found to be faster than expected for words like *type*, a near true friend with a slight difference in pronunciation (Dijkstra et al., 1999; Smits et al., 2006; Dijkstra et al., 2010). Similar cognate facilitation effects also occur in production tasks, such as picture-naming. If an advanced Catalan-Spanish bilingual is asked to name pictures in Spanish, they are faster to do so for true friends such as *gato* (*gat* in Catalan 'cat') than for non-cognates. The effect, although smaller, can also be obtained when pictures are to be named in the L1 (Costa et al., 2000). Similar effects have also been obtained for Japanese-English bilinguals, despite the difference in the scripts (Hoshino and Kroll, 2008).

3 Predictions

In this work, we ask if the structure of cross-lingual word embeddings spaces have the properties that would be expected given human bilingual behaviour. Assuming the distributed, integrated model of the lexicon proposed in the bilingualism literature, the underlying linking hypothesis is that coactivation effects (whether expressed as similarity judgments or measured as reaction times) are the expression of greater or smaller proximity in a multi-dimentional space. On this basis, several hypotheses are proposed. The first hypothesis aims to establish whether cross-lingual word embeddings are sensitive to a bilingual situation and generate an integrated cross-lingual space. Secondly, we test if cross-lingual word embeddings show the shared-translation effect. Finally we test the cross-linguistic competition/priming of lexical forms from the L1 to the L2 language, comparing cross-lingual to monolingual spaces in true friends and false friends scenarios.

We will often talk about a word and its *translation*. By this term, we mean the pair of words that a bilingual dictionary would indicate as equivalent lexical entries, a translations pair.

[3] German-English speakers, compared to monolingual English speakers, are slower in recognising, for example, that the word *bubble* is infelicitous in contexts where the word *blister* is required, due to the fact that these two words are translated as the same word *Blase* in German.

[4] Notice that this effect is robust as it was also replicated for English-Hebrew bilinguals, who learned Hebrew as an L2. Moreover, Degani et al. (2011) used as stimuli semantically unrelated word pairs, extending previously established results for sense-related words, such as *home-house* (Jiang, 2002).

3.1 The integrated bilingual lexicon

We test the idea that the bilingual lexicon is an integrated system by looking at effects of such a system in one-to-one mappings and one-to-many mappings.

The simplest and most basic prediction that a model of the integrated bilingual lexicon needs to be able to confirm is that words in the bilingual lexicon are "closer" to each other than word mappings across two aligned mono-lingual lexicons.

HYPOTHESIS 1 Given an L1 word w_1 and its translation w_2 in L2, the similarity between the word embeddings pair in a cross-lingual space (w_1^{cr}, w_2^{cr}) is higher than their similarity between their aligned monolingual counterparts $(w_1^{m_1}, w_2^{m_2})$.

$$sim(w_1^{cr_1}, w_2^{cr_2}) > sim(w_1^{m_1}, w_2^{m_2}) \quad (1)$$

The second prediction is based on the finding of shared translations, where a shared translation in L1 affects L2 similarity judgments.

HYPOTHESIS 2 Given an L1 word w_1 and its translations w_{2a} and w_{2b} in L2, the similarity between the cross-lingual embeddings of the translation pair will be greater than the similarity between their monolingual counterparts.

$$sim(w_{2a}^{cr_2}, w_{2b}^{cr_2}) > sim(w_{2a}^{m_2}, w_{2b}^{m_2}) \quad (2)$$

3.2 Form-meaning competitions in the biligual lexicon

The following experiments investigate the competition faced by words with a high level of lexical similarity. Simplifying, words across two languages can be similar in form or meaning, or both or neither. For the following predictions, then, we define five different types of word pairs. Examples are shown in Figure 3.

FALSE FRIENDS: words that share the same form, but are semantically different.

REAL TRANSLATIONS of the false friends: the real L2 translations of the L1 word that also has a false friend.

TRUE FRIENDS: words sharing form and meaning.

NORMAL TRANSLATIONS: words semantically equivalent, but with a different form.

UNCORRELATED WORDS: words lexically and semantically uncorrelated.

False and true friends coactivation In bilingual speakers, false friends show an inhibitory effect of the L1 meaning in L2 tasks, but they do not affect the final accuracy of the task completion. Consequently, in our cross-lingual vectorial space, false friends should not have higher similarity score than their real translations, but they should be included in the top translations, i.e. the difference in similarity score between the real translation and the false friends should be smaller than the difference between the real translations and other words (appropriately matched to the false friends). This in turn can be demonstrated by two expected inequalities: false friends are not closer than real translations but false friends are significantly more similar than (matching) uncorrelated words.

Precisely, given an L1 word w_1, and its real L2 translation w_2 and the false friend w_{2ff} in cross-lingual space, we expect the similarity score between the pair (w_1, w_2) not to be lower than the similarity score between the pair (w_1, w_{2ff}).

HYPOTHESIS 3 Real translations have a better or equal similarity score than their corresponding false friends.

$$sim(w_1, w_2) >= sim(w_1, w_{2ff}) \quad (3)$$

Moreover, false friends (w_1, w_{2ff}) have a similarity score that is higher than uncorrelated words (w_1, w_{2nc}). This is because the false friends pair (w_1, w_{2ff}) shares similarity of form even if it is, in fact, semantically uncorrelated.

HYPOTHESIS 4 False friends have a better similarity score than pairs with no correlation.

$$sim(w_1, w_{2ff}) > sim(w_1, w_{2nc}) \quad (4)$$

Lexical similarity can also work in the opposite direction. L1 words that are similar to the L2 word both in form and meaning, *true friends*, have been shown, in bilinguals speakers, to facilitate tasks in L2. In cross-lingual word embeddings, we expect that true friends (w_{1tf}, w_{2tf}) have a higher similarity score than normal translation pairs (w_{1n}, w_{2n}), whose interaction is not enhanced by lexical or morphological resemblances.

HYPOTHESIS 5 True friends have a better similarity score than normal translation pairs.

$$sim(w_{1tf}, w_{2tf}) > sim(w_{1n}, w_{2n}) \quad (5)$$

HYP. 1 Cross-lingual word embeddings pairs are more similar than their aligned monolingual counterparts	$sim(w_1^{cr_1}, w_2^{cr_2}) > sim(w_1^{m_1}, w_2^{m_2})$
HYP. 2 For two L2 words sharing a translation in L1, cross-lingual word embeddings are more similar than monolingual word embeddings	$sim(w_{2a}^{cr_2}, w_{2b}^{cr_2}) > sim(w_{2a}^{m_2}, w_{2b}^{m_2})$
HYP. 3 Real translations are more similar than their corresponding false friends	$sim(w_1, w_2) >= sim(w_1, w_{2ff})$
HYP. 4 False friends are more similar than uncorrelated pairs	$sim(w_1, w_{2ff}) > sim(w_1, w_{2nc})$
HYP. 5 True friends are more similar than normal translation pairs	$sim(w_{1tf}, w_{2tf}) > sim(w_{1n}, w_{2n})$
HYP. 6 Normal translation pairs are more similar than real translations of false friends	$sim(w_{1n}, w_{2n}) > sim(w_1, w_2)$

Figure 1: The six experimental predictions.

Another hypotheses can also be formulated that follows logically from these preceding ones. Consider the pair (w_1, w_2) where w_2, as seen before, is the real translation of w_1 in a pair that also has a false translation. In this case, it is important to remember that w_1 has a false friend w_{2ff}, so we know that accessing w_2 is more effortful since, for a bilingual speaker, w_{2ff} will also be activated. Consequently, we can assume that a normal pair of words (w_{1n}, w_{2n}), a pair of translated words that have no false friend, are closer in space than (w_1, w_2) precisely because (w_{1n}, w_{2n}) is not inhibited by a false translation as in the case of (w_1, w_2).

HYPOTHESIS 6 Normal translation pairs have a higher similarity score than real translations of false friends.

$$sim(w_{1n}, w_{2n}) > sim(w_1, w_2) \quad (6)$$

The predictions are summarised in Figure 1. If confirmed, they give us a fairly detailed view of the structure of the lexicon conceived as a multi-dimensional, integrated multilingual space. In particular, they inform us on the respective importance of formal and meaning properties of words in this cross-lingual similarity space.

4 Experiments 1 and 2

We test our hypotheses using two different cross-lingual word embeddings models. (The numbering of the experiments corresponds to the numbering of the hypotheses.)

One model is VECMAP, a word-level cross-lingual word embedding method, developed by Artetxe et al. (2018), which offers different options, ranging from fully supervised to weakly supervised or unsupervised, the state-of-the-art for bilingual lexicon induction. This method, unlike other models, does not use an existing dictionary for initialization. The values of the word vectors in both the source and the target distribution are sorted, vectors that have similar permutations are identified as possible translations and are used to initialize a dictionary that is then further improved by self-iterative training.

translation pairs	shared translation pairs
wood-legno	legno bosco
wood-bosco	
grade-grado	grado voto
grade-voto	
block-blocco	blocco ceppo
block-ceppo	blocco bloccare
block-bloccare	blocco ostacoalre
block-ostacolare	ceppo bloccare
	ceppo ostacolare

Figure 2: Sample of translation pairs and sample of shared translation pairs used in experiments 1 and 2.

We also test M2VEC, a weakly-supervised, concept-based adversarial model (Wang et al., 2019). This method is based on the idea that languages use similar words to express similar concepts (Søgaard et al., 2015). The adversarial training uses concepts, drawn from Wikipedia, rather than words, to learn competitive cross-lingual word embeddings. The alignments are learnt by a generative adversarial networks (GAN) adapted to the cross-lingual mapping objective.

FALSE FRIENDS		REAL TRANSLATIONS		TRUE FRIENDS		NORMAL TRANSLATIONS		UNCORRELATED PAIRS	
arrange	arrangiare	arrange	disporre	family	famiglia	jam	marmellata	arrange	tagliare
		arrange	sistemare	fantastic	fantastico	january	gennaio	attend	guardare
		arrange	organizzare	future	futuro	journey	viaggio	attic	luna
attend	attendere	attend	frequentare	general	generale	keep	tenere	attitude	canale
		attend	assistere	generation	generazione	kind	tipo	barrack	tazza
bald	baldo	bald	calvo	guide	guida	leave	partire	brave	forchetta
		bald	pelato	historial	storica	light	luce	camera	traduzione
brave	bravo	brave	coraggioso	industry	industria	mean	significare	caution	muro
		brave	valoroso	local	locale	mood	umore	code	pistola
canteen	cantina	canteen	mensa	melody	melodia	overview	panoramica	confetti	vitamine
		canteen	borraccia	minor	minore	pattern	modello	confidence	treno

Figure 3: Sample of five word pair types used in experiments 3, 4, 5, and 6.

4.1 Data

Word-embeddings data For training VECMAP, we use the English-Italian portion of the data used in Artetxe et al. (2018), which is based on the dataset described in Dinu et al. (2015). The English-Italian dataset provided by Dinu et al. (2015) contains 300-dimensional CBOW monolingual word embeddings for a total of 200K words trained on the WacKy crawling corpora.[5] The English word embeddings use 2.8 billion tokens (ukWAC + Wikipedia + BNC) and the Italian word embeddings use 1.6 billion tokens (itWAC). An English-Italian gold standard bilingual dictionary built from Europarl [6] word alignments is also provided with a training set of 5000 entries ordered by English frequency. For M2VEC, 300-dimensional monolingual word embeddings are trained with FastText. The training corpus is taken from a Wikipedia dump.[7] The word embeddings are augmented to include concept-aligned articles extracted from the Linguatools Wikipedia comparable corpus.[8] VECMAP uses word-level embeddings and M2VEC character-level embeddings.

Word lists For testing, we build lists of English words that have multiple translations in Italian. The lists of polysemous words were validated with the support of the Cambridge Bilingual Dictionary[9] and by two bilingual speakers. Some example pairs are shown in Figure 2. Word lists are available as supplementary materials.

[5]https://wacky.sslmit.unibo.it/

[6]https://www.statmt.org/europarl/

[7]https://dumps.wikimedia.org/

[8]http://linguatools.org/tools/corpora/wikipedia-comparable-corpora/

[9]https://dictionary.cambridge.org/dictionary/english-italian/

		Mono lingual	Cross-lingual VECMAP	Cross-lingual M2VEC
E1	Mean	0.02	0.371	0.345
	Variance	0.03	0.044	0.044
	T(104)		−17.059	−15.937
	p		0.000	0.000
E2	Mean	0.153	0.163	0.184
	Variance	0.019	0.025	0.044
	T(109)		−2.050	−2.835
	p		0.021	0.003

Table 1: Results of experiments 1 and 2.

4.2 Results

The results are shown in Table 1. Recall the two hypotheses, hypothesis 1 and hypothesis 2 summarised in Figure 1.

The results confirm both hypotheses for both cross-lingual models and are statistically significant under a one-tailed pairwise t-test with $\alpha = 0.25$. Cross-lingual word embeddings have a higher mean similarity score than aligned monolingual word embeddings. Also, cross-lingual word embeddings, like humans, show a shared-translation effect. Both cross-lingual models show higher mean similarity scores for L2-words that share a common L1 source than the monolingual model.

5 Experiments 3, 4, 5 and 6

The next four experiments test whether the hypotheses concerning true and false friends in word embeddings are confirmed. The numbering of the experiments corresponds to the numbering of the hypotheses.

5.1 Data

Word embedding data In this set of experiments, we use the same data from the previous two experiments (the dataset described in Dinu

et al. (2015)) with the addition of the FastText pre-trained word embeddings for English and Italian (Bojanowski et al., 2017).[10] These publicly available vectors are obtained by a 5-word window, for 300 resulting dimensions, on CommonCrawl and Wikipedia data using the Skip-gram model. Every word is represented as an n-grams of characters, for n training between 3 and 6. Each n-gram is represented by a vector and the sum of these vectors forms the vector representing the given word. So we have three cross-lingual models: two versions of VECMAP, one trained on CBOW (word-level) and the other on FastText (character-level), and the concept-based adversarial model M2VEC, which also uses character-based FastText representations. The inclusion of FastText embeddings is important for these experiments, as the embeddings need to be sensitive to character-level sequences to detect similarity of form in false and true friends. VECMAP was used to obtain the cross-lingual word embeddings. Having two versions of VECMAP, one that use Skip-gram and one that uses CBOW is based on their different performance on rare words. The CBOW model predicts a word from its context and is better in accuracy for frequent words, but encounters problems with rare words, while the Skip-gram model predicts the context from a target word, and so has good representation of rare words or phrases (Mikolov et al., 2013a,c). In the figures they will be indicated, respectively, as Vecmap, FastText and Concepts.

Word lists The data required for the following experiments comprise five lists of word pairs, defined in section 3.2: false friends, real translations, true friends, normal translations and uncorrelated words. Examples are shown in Figure 3 and the complete lists are available in the appendix, with their similarity scores.

These word lists have been constructed from various online resources, adding also words that were found serendipitously by the authors in different texts.[11] The normal translations and uncorrelated word pairs were built by one of the authors.

[10] https://github.com/facebookresearch/fastText

[11] The false friends list was built starting from http://www.lifemilan.it/en/false-friends-a-must-learn-list/ and https://www.reference.tjtaylor.net/false-friends/. The true friends list was started from https://takelessons.com/blog/italian-grammar-cognates-z09.

	Judge 1					
Judge 2	FF	TF	RT	NT	UN	Total
FF	93	1	0	0	0	94
TF	3	127	0	0	0	130
RT	0	2	130	8	1	141
NT	1	3	7	133	0	144
UN	0	0	0	0	97	97
Total	97	133	137	141	98	606

Table 2: Inter-judge partition of the five lists of words, rounded to closest integer. TF = true friends; FF = false friends; RT = real translations; NT = normal translations; UN = uncorrelated.

Normal translations were selected by the bilingual dictionary excluding those that had true or false friends. The uncorrelated words were selected ensuring that they were entirely uncorrelated. They were validated with the help of the Cambridge Bilingual dictionary,[12] where each translation of each pair of words was checked to ensure correctness and complete disjunction in meaning. The online English-Italian dictionary was not comprehensive: not all the meanings were reported, unlike the English-Spanish or English-French dictionaries. Therefore, sometimes a false friend was not reported by the English-Italian dictionary, but was found with the help of the other bilingual dictionaries.

Once the lists were constructed, we run an inter-judge agreement validation. The words were shuffled and the two authors, who master both languages well, classified them in the five types discussed above. Cohen's Kappa was 94.6, showing very high agreement. Fractional numbers were distributed in the few cases of multiple classification by one or both judges. The main source of disagreement were those words that are, at the same time, false friends and true friends depending on the context. The inter-judge agreemeent table is shown in Table 2.

As can be seen in Figure 3, the real translations list (151 pairs of words) is larger than the false friends list (97 pairs of words) due to the different meanings that an L1 word can have in L2.

As for the uncorrelated list, the same L1 words from the false friends list have been used for a direct comparison between the false friends similarity and the uncorrelated words similarity. Since

[12] https://dictionary.cambridge.org/dictionary/english-italian/

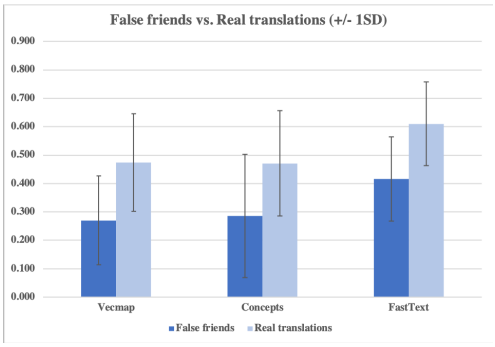


Figure 4: Experiment 3: false friends compared to real translations.

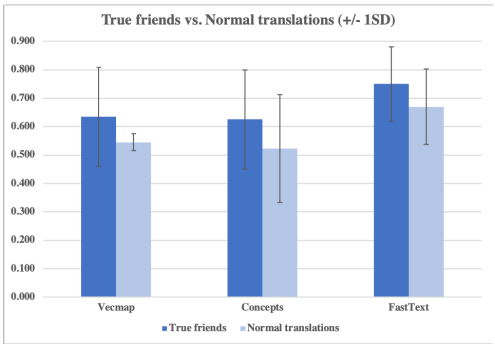


Figure 6: Experiment 5: true friends compared to normal translations.

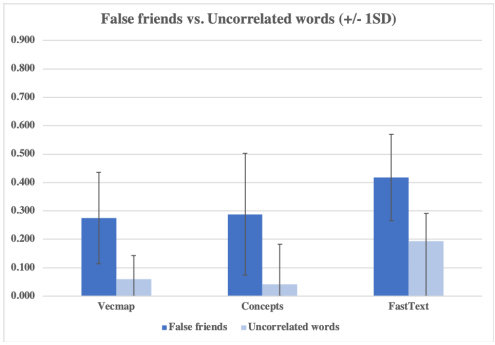


Figure 5: Experiment 4: false friends compared to pairs without correlation.

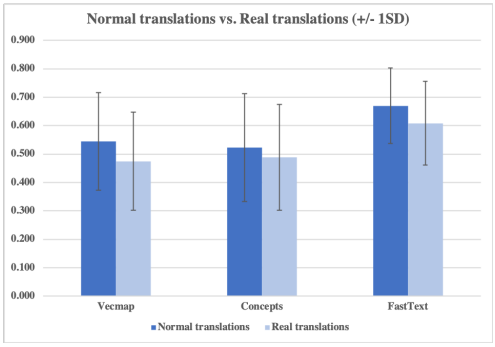


Figure 7: Experiment 6: Normal translations compared to real translations of false friends.

the real translations list contains the largest number of pairs, for the comparison between real translations and normal translations, the real translations have been sampled to have equal size.

5.2 Results

As shown in the Figures 4 to 7, overall the three methods are consistent as they show the same pattern of results. In terms of cosine similarity, the Vecmap+FastText word embeddings tend to have higher means than the other methods.

HYPOTHESIS 3 The first hypothesis concerning false friends —real translations have a better similarity score than their corresponding false friends— is confirmed. Figure 4 shows a cosine similarity that is 0.2 points higher for real translations pairs ($p < 0.0001$ in all three cases for a paired one-tailed t-test).

HYPOTHESIS 4 Based on the scores shown in Figure 5, the second hypothesis concerning false friends is also confirmed. False friends are significantly more similar in the cross-lingual space than uncorrelated words, as the similarity scores for false friends, in all three systems, are higher by more than 0.2 points ($p < 0.0001$ in all three cases).

HYPOTHESIS 5 The hypothesis that true friends have a better similarity score than normal translation pairs is also confirmed. Figure 6 shows that the mean true friends similarity is higher than the mean similarity of the normal translations ($p < 0.001$ in all three cases).

HYPOTHESIS 6 The hypothesis that normal pairs of words have a higher similarity score than real translations of false friends is confirmed, see Figure 7. The mean similarity score for real translations is lower than the mean score for normal translations. This difference is statistically significant for Vecmap and FastText ($p < 0.01$). For the concept-based system, on the other hand, we do not have enough evidence to reject the null hypothesis as $p > 0.025$ and T(296)<t-value. Thus, in this case, we must conclude that the normal translations are as similar as the real translations.

BONFERRONI CORRECTION As Hypotheses 3 and 4, hypotheses 3 and 6 and hypotheses 5 and 6 use the same data (respectively false friends, real translations and normal translations), we run

the Bonferroni correction not to incur in α inflation. All hypotheses are further confirmed, except for hypothesis 6 where the concept-based system again shows a non-significant result (Bonferroni correction higher than α (0.106>0.025)).

6 Discussion

Cross-lingual word embeddings show a lexical structure matching the bilingual lexicon for the three properties that we tested.

They are an integrated multi-lingual space and exhibit the shared-translation effect. This suggests that the initial monolingual embeddings, in our case the Italian ones, are affected by the cross-lingual mapping as word vectors are changed to accommodate cross-lingual interactions. Intuitively, this is similar to the behavior of bilinguals when they gradually learn a new language and semantic links are created by associating the new words to an existing one in their native language.

Cross-lingual word embeddings did not show interference effects of false friends. They are not as affected by the lexical or morphological level when there is a semantic correlation between words. However, false friends show more similarity than uncorrelated words, showing that, like for humans, they have a different status from orthographically and semantically uncorrelated words. True friends in cross-lingual space behave like the lexicon of bilinguals, when the semantic, lexical and morphological level are aligned.

The last hypothesis shows varying results, as only two systems accept the bilingual hypothesis: Vecmap and FastText have a higher similarity score for normal translations, confirming the assumption that real translations are inhibited by false friends. Notice that this result argues again that cross-lingual word embeddings are indirectly affected by false friends, as there is no other difference between real translations and normal translations. Interestingly, on the other hand, the concept-based system is not sensitive to the indirect effect of false friends in the cross-lingual space. This is not unexpected as the concept-based model is less affected by surface form. The difference between word-based models and the concept-based model could yield an hypothesis to be tested for the human lexicon with precise underlying formal models.

Because our predictions are confirmed, they also confirm that the similarity structure defined by current cross-lingual word embedding models is promising as a view of the structure of the lexicon conceived as a multi-dimensional, integrated multilingual space. In particular, our results inform us on the respective importance of formal and meaning properties of words in this cross-lingual similarity space. Notice that the pairwise results described so far define, by transitivity, a total order of similarity: true friends > normal translations > real translations > false friends > uncorrelated pairs. True friends match both in form and meaning, normal and real translations match only in meaning, and false friends match only in form. Thus, this order clearly indicates that while both form and meaning matter, similarity based on meaning is more important that similarity based on form.

7 Related work

The related work for the investigation reported here comprises work on the human bilingual lexicon, cross-lingual word embeddings models and computational models of the bilingual lexicon. As the relevant work on the first topic has already been discussed, we concentrate here on the latter two.

Vectors of words that are semantically or syntactically similar have been shown to be close to each other in the same space (Mikolov et al., 2013a,c; Pennington et al., 2014), making them widely useful in many natural language processing tasks such as machine translation and parsing (Bansal et al., 2014; Mi et al., 2016), both in a single language and across different languages.

Mikolov et al. (2013b) first observed that the geometric positions of similar words in different languages were related by a linear relation. Zou et al. (2013) showed that a cross-lingually shared word embedding space is more useful than a monolingual space in an end-to-end machine translation task. However, traditional methods for mapping two monolingual word embeddings require high quality aligned sentences or dictionaries (Faruqui and Dyer, 2014; Ammar et al., 2016).

Reducing the need for parallel data, then, has become the main issue for cross-lingual word embedding mapping. Methods that rely on sentence-alignments and also document-alignments have been proposed. Hermann and Blunsom (2014) present a method that, given enough data, train bilingual word embeddings from a sentence-

aligned corpus. Luong et al. (2015) propose a model, BiSkip, that takes as input a parallel corpus with both sentence and word-level alignment. Unlike other methods, BiSkip tries to learn not only target word representations from source words but also source word representations from target words. Vulic and Moens (2016) induces bilingual word embeddings from document-aligned comparable data that have been merged and shuffled producing a pseudo-bilingual document.

Some recent work aiming at reducing resources has shown competitive cross-lingual mappings across similar languages, using a pseudo-dictionary, such as identical character strings between two languages (Smith et al., 2017), or a simple list of numerals, thanks to a self-learning iterative framework (Artetxe et al., 2017). Furthermore, as indicated in section 4, Artetxe et al. (2018) extend their self-learning framework to unsupervised models, and build the state-of-the-art for bilingual lexicon induction. Another weakly-supervised model is proposed by Wang et al. (2019), a weakly-supervised concept-based adversarial method, used in our experiments, as also indicated in section 4.

Several computational models of human bilingualism exist, see Li (2013) for an overview. More relatedly to the current work that uses distributional approaches, aspects of the bilingual lexicon have been proposed for word associations (Matusevych et al., 2018). These associations are different in bilingual and monolingual speakers. For example, cognates, collocations and phonological responses are produced more frequently by non-native speakers. This work proposes a model of word association in bilinguals, implemented as a semantic network paired with a retrieval mechanism. Computational models of the influence of the native language on second language learning have also been investigated in Matusevych (2016), specifically for argument structure.

8 Conclusion

In the spirit of better understanding distributed representations and how well they match what we know about the structure and processing of language in humans (Linzen et al., 2016), this paper investigates two models of cross-lingual word embeddings comparing them to the shared translation effect and cross-lingual coactivation effects involving true and false friends found in humans. We find that predictions about cross-lingual word embeddings are mostly confirmed, making them promising functional models of at least some aspects of the bilingual lexicon, despite their structural simplicity.

9 Acknowledgments

We thank Haozhou Wang for giving us access to his model M2VEC and to the embeddings it produces, and to Suzanne Stevenson and Yevgen Matusevych for pointing us to relevant work.

References

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively Multilingual Word Embeddings. *arXiv preprint arXiv:1309.4168*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *55th Annual Meeting of the Association for Computational Linguistics*, pages 451–462, Vancouver, Canada.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia. Association for Computational Linguistics.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring Continuous Word Representations for Dependency Parsing. In *52nd Annual Meeting of the Association for Computational Linguistics*, pages 809–815, Baltimore, Maryland, USA.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Albert Costa, Alfonso Caramazza, and Nuria Sebastian-Galles. 2000. The cognate facilitation effect: Implications for models of lexical access. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 26(5):1283–1296.

Tamar Degani, Anat Prior, and Natasha Tokowicz. 2011. Bidirectional transfer: The effect of sharing a translation. *Journal of Cognitive Psychology*, 23(1):18–28.

Tamar Degani and Natasha Tokowicz. 2010. Ambiguous words are harder to learn. *Bilingualism: Language and Cognition*, 13(3):299314.

Ton Dijkstra, Jonathan Grainger, and Walter J.B. van Heuven. 1999. Recognition of cognates and interlingual homographs: The neglected role of phonology. *Journal of Memory and Language*, 41(4):496 – 518.

Ton Dijkstra, Koji Miwa, Bianca Brummelhuis, Maya Sappelli, and Harald Baayen. 2010. How cross-language similarity and task demands affect cognate recognition. *Journal of Memory and Language*, 62(3):284 – 301.

Ton Dijkstra, Henk Van Jaarsveld, and Sjoerd Ten Brinke. 1998. Interlingual homograph recognition: Effects of task demands and language intermixing. *Bilingualism: Language and Cognition*, 1(1):5166.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *3rd International Conference on Learning Representations*, pages 1–10, Toulon, France.

K. Elston-Güttler and J. N. Williams. 2008. L1 polysemy affects L2 meaning interpretation: evidence for L1 concepts active during L2 reading. *Second Language Research*, 24:167187.

Manaal Faruqui and Chris Dyer. 2014. Improving Vector Space Word Representations Using Multilingual Correlation. In *14th Coference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 58–68, Baltimore, Maryland. Association for Computational Linguistics.

Walter J.B. van Heuven, Ton Dijkstra, and Jonathan Grainger. 1998. Orthographic neighborhood effects in bilingual word recognition. *Journal of Memory and Language*, 39(3):458 – 483.

Noriko Hoshino and Judith F. Kroll. 2008. Cognate effects in picture naming: Does cross-language activation survive a change of script? *Cognition*, 106(1):501 – 511.

Nan Jiang. 2002. Form-meaning mapping in vocabulary acquisition in a second language. *Studies in Second Language Acquisition*, 24(4):617637.

N.F. Johnson and K.R. Pugh. 1994. A cohort model of visual word recognition. *Cognitive Psychology*, 26(3):240 – 346.

Judith F. Kroll and Ton Dijkstra. 2012. The bilingual lexicon. In *The Oxford Handbook of Applied Linguistics*. Oxford University Press.

Ping Li. 2013. Computational modeling of bilingualism: How can models tell us more about the bilingual mind? *Bilingualism: Language and Cognition*, 16(2):241245.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, Denver, Colorado. Association for Computational Linguistics.

Yevgen Matusevych. 2016. *Learning constructions from bilingual exposure. Computational studies of argument structure acquisition*. Ph.D. thesis, Tilburg University.

Yevgen Matusevych, Amir Ardalan Kalantari Dehaghi, and Suzanne Stevenson. 2018. Modeling bilingual word associations as connected monolingual networks. In *Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018)*, pages 46–56, Salt Lake City, Utah. Association for Computational Linguistics.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage Embedding Models for Neural Machine Translation. In *2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, Austin, Texas, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting Similarities among Languages for Machine Translation . *arXiv preprint arXiv:1309.4168*, (Computation and Language):1–10.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013c. Distributed Representations of Words and Phrases and their Compositionality. In *26th International Conference on Neural Information Processing Systems*, pages 3111–3119, Lake Tahoe, Nevada, USA.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove - Global Vectors for Word Representation. In *2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar.

Jennifer Rodd, Gareth Gaskell, and William Marslen-Wilson. 2002. Making sense of semantic ambiguity: Semantic competition in lexical access. *Journal of Memory and Language*, 46(2):245 – 266.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *15th International Conference on Learning Representations*, pages 1–10, Toulon, France.

Erica Smits, Heike Martensen, Ton Dijkstra, and Dominiek Sandra. 2006. Naming interlingual homographs: Variable competition and the role of the decision system. *Bilingualism: Language and Cognition*, 9(3):281297.

Anders Søgaard, Zeljko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual NLP. In *53rd Annual Meeting of the Association for Computational Linguistics and 7th International Joint Conference on Natural Language Processing*, pages 1713–1722, Beijing, China.

Janet G. Van Hell and Annette M. B. de Groot. 1998. Conceptual representation in bilingual memory: Effects of concreteness and cognate status in word association. *Bilingualism: Language and Cognition*, 1(3):193211.

Ivan Vulic and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *J. Artif. Int. Res.*, 55(1):953–994.

Haozhou Wang, James Henderson, and Paola Merlo. 2019. Weakly-supervised concept-based adversarial learning for cross-lingual word embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*, Hong Kong.

Andrea Weber and Anne Cutler. 2004. Lexical competition in non-native spoken-word recognition. *Journal of Memory and Language*, 50(1):1 – 25.

John N. Williams. 2014. The bilingual lexicon. In *The Oxford Handbook of the Word*. Oxford University Press.

Brent Wolter. 2001. Comparing the l1 and l2 mental lexicon. *Studies in Second Language Acquisition*, 23(1):41–69.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual Word Embeddings for Phrase-Based Machine Translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington, USA.

Federated Learning of N-gram Language Models

Mingqing Chen, Ananda Theertha Suresh, Rajiv Mathews, Adeline Wong,
Cyril Allauzen, Françoise Beaufays, Michael Riley
Google, Inc.
{mingqing,theertha,mathews,adelinew,allauzen,fsb,riley}
@google.com

Abstract

We propose algorithms to train production-quality n-gram language models using federated learning. Federated learning is a distributed computation platform that can be used to train global models for portable devices such as smart phones. Federated learning is especially relevant for applications handling privacy-sensitive data, such as virtual keyboards, because training is performed without the users' data ever leaving their devices. While the principles of federated learning are fairly generic, its methodology assumes that the underlying models are neural networks. However, virtual keyboards are typically powered by n-gram language models for latency reasons.

We propose to train a recurrent neural network language model using the decentralized `FederatedAveraging` algorithm and to approximate this federated model server-side with an n-gram model that can be deployed to devices for fast inference. Our technical contributions include ways of handling large vocabularies, algorithms to correct capitalization errors in user data, and efficient finite state transducer algorithms to convert word language models to word-piece language models and vice versa. The n-gram language models trained with federated learning are compared to n-grams trained with traditional server-based algorithms using A/B tests on tens of millions of users of a virtual keyboard. Results are presented for two languages, American English and Brazilian Portuguese. This work demonstrates that high-quality n-gram language models can be trained directly on client mobile devices without sensitive training data ever leaving the devices.



Figure 1: Glide trails are shown for two spatially-similar words: "Vampire" (in red) and "Value" (in orange). Viable decoding candidates are proposed based on context and language model scores.

1 Introduction

1.1 Virtual keyboard applications

Virtual keyboards for mobile devices provide a host of functionalities from decoding noisy spatial signals from tap and glide typing inputs to providing auto-corrections, word completions, and next-word predictions. These features must fit within tight RAM and CPU budgets, and operate under strict latency constraints. A key press should result in visible feedback within about 20 milliseconds (Ouyang et al., 2017; Alsharif et al., 2015). Weighted finite-state transducers have been used successfully to decode keyboard spatial signals using a combination of spatial and language models (Ouyang et al., 2017; Hellsten et al., 2017). Figure 1 shows the glide trails of two spatially-similar words. Because of the similarity of the two trails, the decoder must rely on the language model to discriminate between viable candidates. For memory and latency reasons, especially on low-end devices, the language models are typically based on n-grams and do not exceed ten megabytes. A language model (LM) is a probabilistic model on words. Given previous words $x_1, x_2, \ldots, x_{m-1}$, an LM assigns a probability to the new words, i.e. $p(x_m|x_{m-1}, \ldots, x_1)$. An n-gram LM is a Markovian distribution of order

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 121–130
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

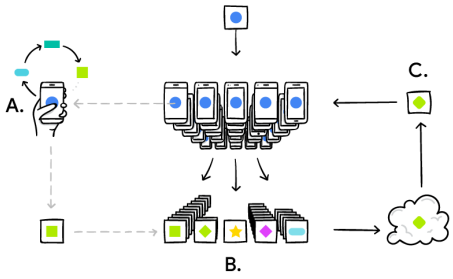


Figure 2: An illustration of the federated learning process from McMahan and Ramage (2017): (A) client devices compute SGD updates on locally-stored data, (B) a server aggregates the client updates to build a new global model, (C) the new model is sent back to clients, and the process is repeated.

$n-1$, defined by

$$p(x_m|x_{m-1},\ldots,x_1) = p(x_m|x_{m-1},\ldots,x_{m-n+1}),$$

where n is the order of the n-gram. For computation and memory efficiency, keyboard LMs typically have higher-order n-grams over a subset of the vocabulary, e.g. the most frequent 64K words, and the rest of the vocabulary only has unigrams. We consider n-gram LMs that do not exceed 1.5M n-grams and include fewer than 200K unigrams.

N-gram models are traditionally trained by applying a smoothing method to n-gram counts from a training corpus (Chen and Goodman, 1999). The highest quality n-gram models are trained over data that are well-matched to the desired output (Moore and Lewis, 2010). For virtual keyboards, training over users' typed text would lead to the best results. Of course, such data are very personal and need to be handled with care.

1.2 Federated learning

We propose to leverage *Federated Learning* (FL) (Konečnỳ et al., 2016; Konečný et al., 2016), a technique where machine learning models are trained in a decentralized manner on end-users' devices, so that raw data never leaves these devices. Only targeted and ephemeral parameter updates are aggregated on a centralized server. Figure 2 provides an illustration of the process. Federated learning for keyboard input was previously explored in Hard et al. (2018), in which a federated recurrent neural network (RNN) was trained for next-word prediction. However, latency constraints prevent the direct use of an RNN for decoding. To overcome this problem, we propose to derive an n-gram LM from a federated RNN LM model and use that n-gram LM for decoding. Specifically, the approximation algorithm is based on `SampleApprox`, which was recently proposed in Suresh et al. (2019a,b). The proposed approach has several advantages:

Improved model quality: Since the RNN LM is trained directly on domain-matched user data, its predictions are more likely to match actual user behavior. In addition, as shown in Suresh et al. (2019a), an n-gram LM approximated from such an RNN LM is of higher quality than an n-gram LM trained on user data directly.

Minimum information transmission: In FL, only the minimal information necessary for model training (the model parameter deltas) is transmitted to centralized servers. The model updates contain much less information than the complete training data.

Additional privacy-preserving techniques: FL can be further combined with privacy-preserving techniques such as secure multi-party computation (Bonawitz et al., 2017) and differential privacy (McMahan et al., 2018; Agarwal et al., 2018; Abadi et al., 2016). By the post-processing theorem, if we train a single differentially private recurrent model and use it to approximate n-gram models, all the distilled models will also be differentially private with the same parameters (Dwork et al., 2014).

For the above reasons, we have not proposed to learn n-gram models directly using `FederatedAveraging` of n-gram counts for all orders.

2 Outline

The paper is organized along the lines of challenges associated with converting RNN LMs to n-gram LMs for virtual keyboards: the feasibility of training neural models with a large vocabulary, inconsistent capitalization in the training data, and data sparsity in morphologically rich languages. We elaborate on each of these challenges below.

Large vocabulary: Keyboard n-gram models are typically based on a carefully hand-curated vocabulary to eliminate misspellings, erroneous capitalizations, and other artifacts. The vocabulary size often numbers in the hundreds of thousands. However, training a neural model directly over the vocabulary is memory intensive as the embedding and softmax layers require space $|\mathcal{V}| \times N_e$, where

$|\mathcal{V}|$ is the vocabulary size and N_e is the embedding dimension. We propose a way to handle large vocabularies for federated models in Section 3.

Incorrect capitalization: In virtual keyboards, users often type with incorrect casing (e.g. "She lives in new york" instead of "She lives in New York"). It would be desirable to decode with the correct capitalization even though the user-typed data may be incorrect. Before the discussion of capitalization, the `SampleApprox` algorithm is reviewed in Section 4. We then modify `SampleApprox` to infer capitalization in Section 5.

Language morphology: Many words are composed of root words and various morpheme components, e.g. "crazy", "crazily", and "craziness". These linguistic features are prominent in morphologically rich languages such as Russian. The presence of a large number of morphological variants increases the vocabulary size and data sparsity ultimately making it more difficult to train neural models. Algorithms to convert between word and word-piece models are discussed in Section 6.

Finally, we compare the performance of word and word-piece models and present the results of A/B experiments on real users of a virtual keyboard in Section 7.

3 Unigram distributions

Among the 200K words in the vocabulary, our virtual keyboard models only use the top 64K words in the higher-order n-grams. We train the neural models only on these most frequent words and train a separate unigram model over the entire vocabulary. We interpolate the two resulting models to obtain the final model for decoding.

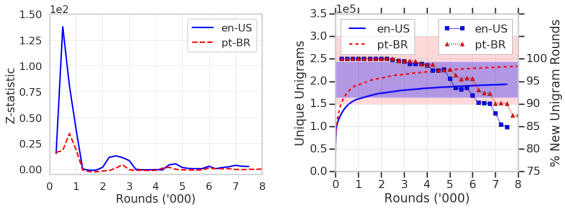
3.1 Collection

Unigrams are collected via a modified version of the `FederatedAveraging` algorithm. No models are sent to client devices. Instead of returning gradients to the server, counting statistics are compiled on each device and returned. In our experiments, we aggregate over groups of approximately 500 devices per training round. We count a unigram distribution U from a whitelist vocabulary by $U = \sum_i w_i C_i$, where i is the index over devices, C_i are the raw unigram counts collected from a single device i, and w_i is a weight applied to device i.

To prevent users with large amounts of data from dominating the unigram distribution, we apply a form of L1-clipping:

$$w_i = \frac{\lambda}{\max(\lambda, \sum C_i)}, \tag{1}$$

where λ is a threshold that caps each device's contribution. When $\lambda = 1$, L1-clipping is equivalent to equal weighting. The limit $\lambda \to \infty$ is equivalent to collecting the true counts, since $w_i \to 1$.



(a) (b)

Figure 3: Unigram distribution convergence. Note that by 3000 rounds, the unigram distribution is stable, but the model is still learning new tail unigrams.

3.2 Convergence

Convergence of the unigram distribution is measured using the unbiased chi-squared statistic (for simplicity, referred to as the Z-statistic) defined in Bhattacharya and Valiant (2015), the number of unique unigrams seen, and a moving average of the number of rounds needed to observe new unigrams.

Figure 3(a) shows the overall distributional convergence based on the Z-statistic. At round k, unigram counts after $k/2$ and k rounds are compared. Figure 3(b) plots the number of whitelist vocabulary words seen and a moving average of the number of rounds containing new unigrams. New unigrams are determined by comparing a round k with all rounds through $k-1$ and noting if any new words are seen. The shaded bands range from the LM's unigram capacity to the size of the whitelist vocabulary.

3.3 Experiments

Since the whitelist vocabulary is uncased, capitalization normalization is applied based on an approach similar to Section 5. We then replace the unigram part of an n-gram model with this distribution to produce the final LM.

In A/B experiments, unigram models with different L1-clipping thresholds are compared against a baseline unigram model gathered from

Model	acc@1 [%]	OOV rate [%]
baseline	8.14	18.08
$\lambda = 1$	$+0.19 \pm 0.21$	-1.33 ± 0.75
$\lambda = 1K$	$+0.11 \pm 0.24$	-1.06 ± 0.66
$\lambda = 5K$	-0.08 ± 0.26	-0.78 ± 0.93

Table 1: Relative change with L1-clipped unigrams on live traffic of en_US users on the virtual keyboard. Quoted 95% confidence intervals are derived using the jackknife method with user buckets.

centralized log data. Results are presented in Table 1. Accuracy is unchanged and OOV rate is improved at $\lambda = 1$ and $\lambda = 1K$.

Before we discuss methods to address inconsistent capitalization and data sparsity in morphologically rich languages, we review `SampleApprox`.

4 Review of `SampleApprox`

`SampleApprox`, proposed in Suresh et al. (2019a,b), can be used to approximate a RNN as a weighted finite automaton such as an n-gram model. A *weighted finite automaton* (WFA) $A = (\Sigma, Q, E, i, F)$ over $\mathbb{R}_+$ (probabilities) is given by a finite alphabet Σ (vocabulary words), a finite set of states Q (n-gram contexts), an initial state $i \in Q$ (sentence start state), a set of final states $F \in Q$ (sentence end states), and a set of labeled transitions E and associated weights that represent the conditional probability of labels (from Σ) given the state (list of n-grams and their probabilities). WFA models allow a special *backoff* label φ for succinct representation as follows. Let $L[q]$ be the set of labels on transitions from state q. For $x \in L[q]$, let $w_q[x]$, be the weight of the transition of x at state q and $d_q[x]$ be the destination state. For a label x and a state q,

$$\begin{aligned} p(x|q) &= w_q[x] && \text{if } x \in L[q], \\ &= w_q[\varphi] \cdot p(x|d_q[\varphi]) && \text{otherwise.} \end{aligned}$$

In other words, φ is followed if $x \notin L[q]$. The definition above is consistent with that of backoff n-gram models (Chen and Goodman, 1999). Let $B(q)$ denote the set of states from which q can be reached by a path of backoff labels and let $q[x]$ be the first state at which label x can be read by following a backoff path from q.

Given an unweighted finite automaton A and a neural model, `SampleApprox` finds the probability model on A that minimizes the Kullback-Leibler (KL) divergence between the neural model and the WFA. The algorithm has two steps: a counting step and a KL minimization step. For the counting step, let $\bar{x}(1), \bar{x}(2), \ldots, \bar{x}(k)$ be k independent samples from the neural model. For a sequence $\bar{x}$, let x_i denote the i^{th} label and $\bar{x}^i = x_1, x_2, \ldots, x_i$ denote the first i labels. For every $q \in Q$ and $x \in \Sigma$, the algorithm computes $C(x, q)$ given by

$$\sum_{q' \in B(q)} \sum_{j=1}^{m} \sum_{i \geq 0} 1_{q(\bar{x}^i(j))=q', q=q'[x]} \cdot p_{\text{nn}}(x|\bar{x}^i(j)).$$

We illustrate this counting with an example. Suppose we are interested in the count of the bi-gram New York. Given a bi-gram LM, `SampleApprox` generates m sentences and computes

$$C(\text{York}, \text{New}) = \sum_{j, i: x_i(j)=\text{New}} p_{\text{nn}}(\text{York}|\bar{x}^i(j)).$$

In other words, it finds all sentences that have the word New, observes how frequently York appears subsequently, and computes the conditional probability. After counting, it uses a difference of convex (DC) programming based algorithm to find the KL minimum solution. If ℓ is the average number of words per sentence, the computational complexity of counting is $\tilde{\mathcal{O}}(k \cdot \ell \cdot |\Sigma|)$ [1] and the computational complexity of the KL minimization is $\tilde{\mathcal{O}}(|E| + |Q|)$ per iteration of DC programming.

5 Capitalization

As mentioned in Section 2, users often type with incorrect capitalization. One way of handling incorrect capitalization is to store an on-device capitalization normalizer (Beaufays and Strope, 2013) to correctly capitalize sentences before using them to train the neural model. However, capitalization normalizers have large memory footprints and are not suitable for on-device applications. To overcome this, the neural model is first trained on uncased user data. `SampleApprox` is then modified to approximate cased n-gram models from uncased neural models.

As before, let $\bar{x}(1), \bar{x}(2), \ldots, \bar{x}(k)$ be k independent (uncased) samples from the neural model. We capitalize them correctly at the server using Beaufays and Strope (2013). Let

[1] $a_n = \tilde{\mathcal{O}}(b_n)$, means $a_n \leq b_n \cdot \text{poly}\log(n), \forall n \geq n_0$.

$\bar{y}(1), \bar{y}(2), \ldots \bar{y}(k)$ represent the corresponding k correctly capitalized samples. Let p_{cap} be another probability model on non-user data that approximates the ratio of uncased to cased probabilities given a context. Given a label y, let $u(y)$ be the uncased symbol. For example, if y is York, then $u(y)$ is york. With the above definitions, we modify the counting step of `SampleApprox` as follows:

$$\sum_{q' \in B(q)} \sum_{j=1}^{m} \sum_{i \geq 0} 1_{q(\bar{y}^i(j))=q', q=q'[y]} \cdot \tilde{p}(y|\bar{y}^i(j)),$$

where $\tilde{p}(y|\bar{y}^i(j))$ is given by

$$p_{\text{nn}}(u(y)|u(\bar{y}^i(j))) \cdot \frac{p_{\text{cap}}(y|\bar{y}^i(j))}{\sum_{y':u(y')=u(y)} p_{\text{cap}}(y'|\bar{y}^i(j))}.$$

We refer to this modified algorithm as `CapSampleApprox`. We note that word-piece to word approximation incurs an additional computation cost of $\tilde{\mathcal{O}}((|E|+|Q|+|\Delta|)\ell)$, where Δ is the number of words, E and Q are the set of arcs and set of states in the word n-gram model, and ℓ is the maximum number of word-pieces per word.

6 Morphologically rich languages

To train neural models on morphologically rich languages, subword segments such as byte-pair encodings or word-pieces (Shibata et al., 1999; Schuster and Nakajima, 2012; Kudo, 2018) are typically used. This approach assigns conditional probabilities to subword segments, conditioned on prior subword segments. It has proved successful in the context of speech recognition (Chiu et al., 2018) and machine translation (Wu et al., 2016). Following these successes, we propose to train RNN LMs with word-pieces for morphologically rich languages.

We apply the word-piece approach of Kudo (2018), which computes a word-piece unigram LM using a word-piece inventory $\mathcal{V}_{\mathcal{P}}$. Each word-piece $x_i \in \mathcal{V}_{\mathcal{P}}$ is associated with a unigram probability $p(x_i)$. For a given word y and its possible segmentation candidates, the word is encoded with the segmentation that assigns the highest probability.

Throughout this paper we apply 4K, 16K, and 30K as the word-piece inventory sizes. These values lie within a range that provides good trade-off between the LSTM embedding size and the richness of the language morphology. We apply 100% character coverage to include all the symbols that appeared in the unigram distribution (Section 3), including the common English letters, accented letters e.g. é, ô, and digits. Accented letters are important for languages like Portuguese. For fast decoding, the n-gram models still need to be at the word-level, since word-piece n-gram models increase the depth of the beam-search during decoding. We convert the word n-gram topology to an equivalent word-piece WFA topology and use `SampleApprox` to approximate the neural word-piece model on the word-piece WFA topology. We then convert the resulting word-piece WFA LM to the equivalent n-gram LM. The remainder of this section outlines efficient algorithms for converting between word and word-piece WFA models.

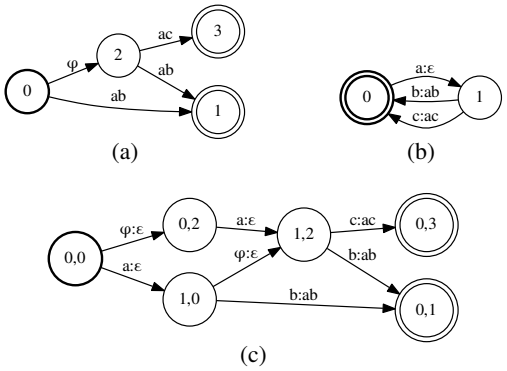


Figure 4: The (a) WFA A and WFSTs (b) T and (c) B for the word vocabulary $\{ab, ac\}$ and word-piece vocabulary $\{a, b, c\}$. Initial states are represented by bold circles and final states by double circles.

A natural way to represent the transduction from word-piece sequences to word sequences is with a finite-state transducer. Given the properties of our word-piece representation, that transducer can be made sequential (i.e., input deterministic).

A *sequential weighted finite-state transducer* (WFST) is a deterministic WFA where each transition has an output label in addition to its (input) label and weight. We will denote by $o_q[x]$ the output label of the transition at state q with input label x, $o_q[x] \in \Delta \cup \{\epsilon\}$, where Δ denotes the output alphabet of the transducer and ϵ the empty string/sequence.

Let M be the minimal sequential (unweighted) finite-state transducer (FST) lexicon from word-piece sequences in Σ^* to word sequences in Δ^*, where Σ denotes our word-piece inventory, Δ denotes our vocabulary, and $*$ is Kleene closure. A word-piece topology B equivalent to the word

Algorithm 1 Approximating a Neural Model as an N-Gram with a Supplemental Topology.

Train R^u_W, R^u_P with `FederatedAveraging`[a]
Train A_W from supplemental corpus $\mathbb{C}$
A_{W_e},A_{W_i},A_{W_m},$A_{W_r} \leftarrow$ Gen(R^u_W, A_W, $\emptyset$, NN2WFA W)
A_{P_e},A_{P_i},A_{P_m},$A_{P_r} \leftarrow$ Gen(R^u_P, A_W, A_{W_i}, NN2WFA P)

function Gen(R^u, A_W, A_{W_i}, function NN2WFA)
  $A_e \leftarrow$ NN2WFA (R^u, A_W)
  if NN2WFA ==NN2WFA W **then**
    $A_i \leftarrow$ NN2WFA (R^u, A_W, self_infer=true)
  else
    $A_i \leftarrow$ NN2WFA (R^u, A_{W_i})
  end if
  $A_m \leftarrow$ Interpolate(A_e, A_i)
  $A_r \leftarrow$ NN2WFA (R^u, A_m)
  return A_e, A_i, A_m, A_r
end function
function NN2WFA W(R^u_W, A_W, self_infer=false)
  if self_infer **then**
    return `CapSampleApprox` (R^u_W, $\emptyset$, A_W)
  else
    return `CapSampleApprox` (R^u_W, A_W, A_W)
  end if
end function
function NN2WFA P(R^u_P, A_W)
  $T^u_W \leftarrow$ ConvertToLowercaseTopology(A_W)
  $T^u_P \leftarrow$ ConvertToWordPieceTopology(T^u_W)
  $A^u_P \leftarrow$ `SampleApprox` (R^u_P, T^u_P)
  $A^u_W \leftarrow$ ConvertToWordTopology(A^u_P)
  return `CapSampleApprox` (A^u_W, A_W, A_W)
end function

[a] T denotes an unweighted topology and A denotes the weighted n-gram model. Superscript u represents uncased models.

topology A can be obtained by composing the word-piece-to-word transducer M with A:

$$B = M \circ A.$$

Since A has backoff transitions, the generic composition algorithm of (Allauzen et al., 2011) is used with a custom composition filter that ensures the result, B, is deterministic with a well-formed backoff structure, and hence is suitable for the counting step of `SampleApprox`. We give an explicit description of the construction of B, from which readers familiar with Allauzen et al. (2011) can infer the form of the custom composition filter.

The states in B are pairs (q_1, q_2), with $q_1 \in Q_M$ and q_2 in Q_A, initial state $i_B = (i_M, i_A)$, and final state $f_B = (f_M, f_A)$. Given a state $(q_1, q_2) \in Q_B$, the outgoing transitions and their destination states are defined as follows. If $x \in L[q_1]$, then an x-labeled transition is created if one of two conditions holds:

1. if $o_{q_1}[x] \in L[q_2]$, then

$$d_{(q_1,q_2)}[x] = (d_{q_1}[x], d_{q_2}[o_{q_1}[x]]) \text{ and}$$
$$o_{(q_1,q_2)}[x] = o_{q_1}[x];$$

2. if $o_{q_1}[x] = \epsilon$ and $R[d_{q_1}[x]] \cap L[q_2] \neq \emptyset$, then

$$d_{(q_1,q_2)}[x] = (d_{q_1}[x], d_{q_2}[o_{q_1}[x]]) \text{ and}$$
$$o_{(q_1,q_2)}[x] = \epsilon$$

where $R[q]$ denotes the set of output non-ϵ labels that can be emitted after following an output-ϵ path from q. Finally if $\varphi \in L[q_1]$, a backoff transition is created:

$$d_{(q_1,q_2)}[\varphi] = (q_1, d_{q_2}[\varphi]) \text{ and } o_{q_1,q_2}[\varphi] = \epsilon.$$

The counting step of `SampleApprox` is applied to B, and transfers the computed counts from B to A by relying on the following key property of M. For every word y in Δ, there exists a unique state $q_y \in Q_M$ and unique word-piece x_y in Σ such that $o_{q_y}[x_y] = y$. This allows us to transfer the counts from B to A as follows:

$$w_q[y] = w_{(q_y,q)}[x_y]$$

The KL minimization step of `SampleApprox` to A is applied subsequently.

As an alternative, the unweighted word automaton A could be used to perform the counting step directly. Each sample $\bar{x}(j)$ could be mapped to a corresponding word sequence $\bar{y}(j)$, mapping out-of-vocabulary word-piece sequences to an unknown token. However, the counting steps would have become much more computationally expensive, since $p_{\text{nn}}(y|\bar{y}^i(j))$ would have to be evaluated for all i, j and for all words y in the vocabulary, where p_{nn} is now a word-piece RNN.

7 Experiments

7.1 Neural language model

LSTM models (Hochreiter and Schmidhuber, 1997) have been successfully used in a variety of sequence processing tasks. LSTM models usually have a large number of parameters and are not suitable for on-device learning. In this work, we use various techniques to reduce the memory footprint and to improve model performance.

We use a variant of LSTM with a Coupled Input and Forget Gate (CIFG) (Greff et al., 2017) for the federated neural language model. CIFG couples

Model	N_l	N_h	N_e	S_e	S_{total}
W_{30K}	1	670	96	2.91M	3.40M
$P_{4K\text{-}S}$	1	670	96	0.38M	0.85M
$P_{4K\text{-}L}$	2	1080	140	0.56M	2.70M
$P_{4K\text{-}G}$	2	1080	280	1.12M	2.71M
$P_{16K\text{-}S}$	1	670	96	1.54M	2.00M
$P_{16K\text{-}L}$	1	670	160	2.56M	3.33M
P_{30K}	1	670	96	2.91M	3.40M

Table 2: Parameters for neural language models. W and P refer to word and word-piece models, respectively. N_l, N_h, N_e, S_e and S_{total} refer to the number of LSTM layers, the number of hidden states in LSTM, the embedding dimension size, the number of parameters in the embedding layer and in total, respectively. The suffixes "S" and "L" indicate small and large models. "G" represents GLSTM. The suffixes 4K, 16K and 30K represent the vocabulary sizes.

the forget and input decisions together, which reduces the number of LSTM parameters by 25%. We also use group-LSTM (GLSTM) (Kuchaiev and Ginsburg, 2017) to reduce the number of trainable variables of an LSTM matrix by the number of feature groups, k. We set $k = 5$ in experiments. Table 2 lists the parameter settings of the word (W) and word-piece (P) models used in this study. Due to the memory limitations of on-device training, all models use fewer than $3.5M$ parameters. For each vocabulary size, we first start with a base architecture consisting of one LSTM layer, a 96-dimensional embedding, and 670 hidden state units. We then attempt to increase the representational power of the LSTM cell by increasing the number of hidden units and using multi-layer LSTM cells (Sutskever et al., 2014). Residual LSTM (Kim et al., 2017) and layer normalization (Lei Ba et al., 2016) are used throughout experiments, as these techniques were observed to improve convergence. To avoid the restriction that $N_h = N_e$ in the output, we apply a projection step at the output gate of the LSTM (Sak et al., 2014). This step reduces the dimension of the LSTM hidden state from N_h to N_e. We also share the embedding matrix between the input embedding and output softmax layer, which reduces the memory requirement by $|\mathcal{V}| \times N_e$. We note that other recurrent neural models such as *gated recurrent units* (Chung et al., 2014) can also be used instead of CIFG LSTMs.

The federated RNN LMs are trained on two language settings of the virtual keyboard: American English (en_US) and Brazilian Portuguese (pt_BR). Following McMahan et al. (2017), 500 reporting clients are used to compute the gradient updates for each round. A server-side learning rate of 1.0, a client-side learning rate of 0.5, and Nesterov momentum of 0.9 are used. Both the word and word-piece models are trained over the same time range and with the same hyperparameters. Prior to federated training of the RNN LM, the word-piece inventory is constructed from the unigram distribution collected via the federated approach introduced in Section 3.

A common evaluation metric for both word and word-piece models is desirable during federated training. Such a metric can be used to monitor the training status and select models to be used for the `CapSampleApprox` algorithm. Neither cross-entropy nor accuracy serves this need due to the mismatch in vocabularies used. Word-level accuracy is hard to compute for the word-piece model, since it requires hundreds of inference calls to traverse all combinations of a word from the word-piece vocabulary. In this study, we apply sentence log likelihood (SLL) in the evaluation. Given a sentence $\bar{x}^m = \{x_1, x_2, \ldots, x_m\}$ composed of m units (either words or word-pieces), SLL is evaluated as $\sum_{i=1}^{m} \log(p_{nn}(x_i|\bar{x}^{i-1}))$. One issue that arises is the handling of out-of-vocabulary (OOV) words. The OOV probability of the word model is about 8%. The comparable probability of an OOV word (according to $\mathcal{V}$) for word-piece models is the product of the corresponding word-piece conditional probabilities, which is much smaller than 8%. To mitigate this issue, we define SLL excluding OOV as:

$$\text{SLL}^e = \sum_{i:x_i \neq \text{OOV}}^{m} \log(p_{nn}(x_i|\bar{x}^{i-1})),$$

where the OOV in the equation includes word-pieces that are components of OOV words. In the following, SLL^e is used as model selection metric.

7.2 Approximated n-gram model

Algorithm 1 illustrates the workflow we use to generate different n-gram models for evaluation. Recall that `CapSampleApprox` takes a RNN LM, an n-gram topology, and a reweighting FST for capitalization normalization. The n-gram topology is empty under self-inference mode. Suresh et al. (2019a) showed that inferring topology from the RNN LM does not perform as well as

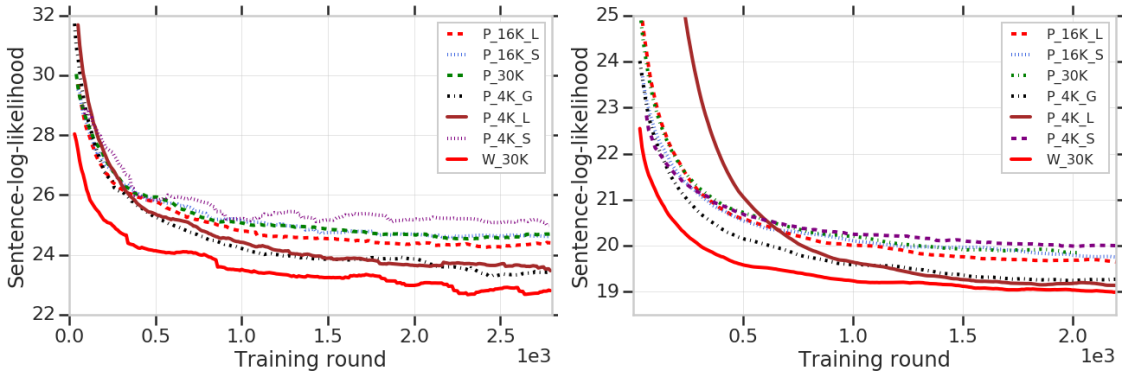


Figure 5: Sentence log likelihood excluding OOV token for en_US (left) and pt_BR (right).

Model	en_US	pt_BR
Baseline	10.03%	8.55%
A_{W_e}	$10.52 \pm 0.03\%$	$9.66 \pm 0.02\%$
A_{W_i}	$10.47 \pm 0.02\%$	$9.67 \pm 0.02\%$
A_{W_m}	$10.27 \pm 0.03\%$	$9.40 \pm 0.02\%$
A_{W_r}	$10.49 \pm 0.03\%$	$9.65 \pm 0.02\%$

Table 3: Result of top-1 prediction accuracy on the live traffic of the virtual keyboard for en_US and pt_BR populations. Quoted 95% confidence intervals for federated models are derived using the jackknife method.

Model	top-1
Baseline	10.03%
A_{P_e}	$10.49 \pm 0.03\%$
A_{P_i}	$10.46 \pm 0.03\%$
A_{P_m}	$10.48 \pm 0.04\%$
A_{P_r}	$10.53 \pm 0.03\%$

Table 4: Result of top-1 prediction accuracy on the live traffic of the virtual keyboard for en_US derived using word-piece models.

using the true n-gram topology obtained from the training corpus. Hence, we supplement the neural-inferred topology with the topology obtained by a large external large corpus denoted by A_W. We use `CapSampleApprox` on four topologies and compare the resulting models: an n-gram model obtained from an external corpus's topology A_e, an n-gram model obtained from a neural inferred topology A_i, an n-gram model obtained by interpolating (merging) the two models above A_m, and an n-gram model obtained by approximating on the interpolated topology A_r. We repeat this experiment for both word and word-piece RNN LMs and use subscripts W and P, respectively. We evaluate all eight produced n-gram models directly on the traffic of a production virtual keyboard, where prediction accuracy is evaluated over user-typed words.

7.3 Results

Figure 5 shows the SLL^e metric for all the experiments listed in Table 2. In general, larger models generate better results than smaller baseline models. For the baseline architectures with same RNN size, having a larger vocabulary leads to some gains. For the larger architectures that have similar total numbers of parameters, 4K word-piece models are shown to be superior to 16K and 30K. For 4K word-piece models, GLSTM is in general on-par with its $P_{4K\text{-}L}$ counterpart. The word model is better than all the word-piece models in both languages in SLL^e. We were surprised by this result, and hypothesize that it is due to the SLL^e metric discounting word-piece models' ability to model the semantics of OOV words. The solid lines are the best models we pick for A/B experiment evaluation for the virtual keyboard ($P_{4K\text{-}L}$ and W_{30K}).

Table 3 shows the A/B evaluation result on both en_US and pt_BR populations. The baseline model is an n-gram model trained directly from centralized logs. All of the federated trained models perform better than the baseline model. We repeated the A/B evaluation with word-piece models on en_US and the results are in Table 4. The performance of word-piece models is similar to that of word models. Among the federated models for en_US, A_{P_r} has the best result. This meets our expectation that the supplemental corpus helps improve the performance of the topology inferred from the RNN LM.

8 Conclusion

We have proposed methods to train production-quality n-gram language models using federated learning, which allows training models without user-typed text ever leaving devices. The proposed methods are shown to perform better than traditional server-based algorithms in A/B experiments on real users of a virtual keyboard.

Acknowledgments

The authors would like to thank colleagues in Google Research for providing the federated learning framework and for many helpful discussions.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM.

Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and Brendan McMahan. 2018. cpsgd: Communication-efficient and differentially-private distributed sgd. In *Neural Information Processing Systems*.

Cyril Allauzen, Michael Riley, and Johan Schalkwyk. 2011. A filter-based algorithm for efficient composition of finite-state transducers. *International Journal of Foundations of Computer Science*, 22(8):1781–1795.

Ouais Alsharif, Tom Ouyang, Françoise Beaufays, Shumin Zhai, Thomas Breuel, and Johan Schalkwyk. 2015. Long short term memory neural network for keyboard gesture decoding. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2076–2080.

Françoise Beaufays and Brian Strope. 2013. Language model capitalization. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6749–6752. IEEE.

Bhaswar Bhattacharya and Gregory Valiant. 2015. Testing closeness with unequal sized samples. In *Advances in Neural Information Processing Systems 28*. NIPS.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1175–1191, New York, NY, USA. ACM.

Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2017. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604.

Lars Hellsten, Brian Roark, Prasoon Goyal, Cyril Allauzen, Francoise Beaufays, Tom Ouyang, Michael Riley, and David Rybach. 2017. Transliterated mobile keyboard input via weighted finite-state transducers. In *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing (FSMNLP)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. 2017. Residual LSTM: design of a deep recurrent architecture for distant speech recognition. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 1591–1595.

Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*.

Oleksii Kuchaiev and Boris Ginsburg. 2017. Factorization tricks for LSTM networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 66–75.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 1273–1282.

Brendan McMahan and Daniel Ramage. 2017. Federated learning: Collaborative machine learning without centralized training data. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning differentially private recurrent language models. In *International Conference on Learning Representations (ICLR)*.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 220–224, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tom Ouyang, David Rybach, Françoise Beaufays, and Michael Riley. 2017. Mobile keyboard input decoding with finite-state transducers. *CoRR*, abs/1704.03987.

Haşim Sak, Andrew Senior, and Françoise Beaufays. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. 1999. Byte pair encoding: A text compression scheme that accelerates pattern matching. Technical report, Technical Report DOI-TR-161, Department of Informatics, Kyushu University.

Ananda Theertha Suresh, Michael Riley, Brian Roark, and Vlad Schogol. 2019a. Approximating probabilistic models as weighted finite automata. *CoRR*, abs/1905.08701.

Ananda Theertha Suresh, Brian Roark, Michael Riley, and Vlad Schogol. 2019b. Distilling weighted finite automata from arbitrary probabilistic models. In *Proceedings of the 14th International Conference on Finite State Methods and Natural Language Processing (FSMNLP 2019)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Learning Conceptual Spaces with Disentangled Facets

Rana Alshaikh
School of CS & Informatics
Cardiff University, UK
alshaikhr@cardiff.ac.uk

Zied Bouraoui
CRIL - CNRS & Univ Artois
France
zied.bouraoui@cril.fr

Steven Schockaert
School of CS & Informatics
Cardiff University, UK
schockaerts1@cardiff.ac.uk

Abstract

Conceptual spaces are geometric representations of meaning that were proposed by Gärdenfors (2000). They share many similarities with the vector space embeddings that are commonly used in natural language processing. However, rather than representing entities in a single vector space, conceptual spaces are usually decomposed into several facets, each of which is then modelled as a relatively low-dimensional vector space. Unfortunately, the problem of learning such conceptual spaces has thus far only received limited attention. To address this gap, we analyze how, and to what extent, a given vector space embedding can be decomposed into meaningful facets in an unsupervised fashion. While this problem is highly challenging, we show that useful facets can be discovered by relying on word embeddings to group semantically related features.

1 Introduction

Conceptual spaces (Gärdenfors, 2000) are vector space models that are aimed at representing the entities of a given kind (e.g. movies), together with their associated properties (e.g. scary) and concepts (e.g. thrillers). As such, they are similar in spirit to the vector space models that have been proposed in information retrieval (Deerwester et al., 1990) and natural language processing (Turney and Pantel, 2010; Mikolov et al., 2013), but there are also notable differences. First, in the context of conceptual spaces, an explicit distinction is made between the entities from the domain of discourse, which are represented as vectors, and the corresponding properties and concepts, which are represented as regions (e.g. polytopes) or soft regions (e.g. characterized by a Gaussian). Second, conceptual spaces are organised into a set of facets[1], each of which captures a different aspect of meaning. For instance, in a conceptual space of movies, we may have facets such as genre, language, geographic location, etc.

Each facet is associated with its own vector space, which intuitively captures similarity w.r.t. the corresponding facet. For instance, in a conceptual space of movies, the vector space for the budget facet would only capture whether two movies had a similar budget. Most of these facet spaces tend to be low-dimensional (e.g. modelling budget only needs a single dimension). This clearly differentiates them from traditional semantic spaces, which often have hundreds of dimensions. From an application point-of-view, the separation of vector space models into facets is appealing for several reasons. One key advantage is that it allows us to model similarity in a more flexible, and cognitively more plausible way. A related advantage is that the low-dimensional nature of the facet-specific spaces should make it easier to learn from few examples. Finally, the separation into facets can also make conceptual spaces more interpretable. However, the study of conceptual spaces has mostly focused on modelling cognitive and linguistic phenomena, such as metaphor (Gärdenfors, 1996) and vagueness (Douven et al., 2013), with only few works addressing the challenge of learning such representations from data.

Decomposing conceptual spaces into facets is similar to the problem of disentangled representation learning (DRL), which has recently received considerable interest. However, empirical studies suggest that purely unsupervised DRL methods are unlikely to be successful without a strong inductive bias. In fact, Locatello et al. (2018) found that what mostly matters was how such methods are initialized, rather than what particular optimization objective is used. Moreover, much of

[1] These facets are often referred to as *domains* in the context of conceptual spaces. However, we will use the term facets to avoid confusion with domains of discourse.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 131–139
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

the work in DRL has focused on image processing rather than textual data (which is what we use in this paper). Finally, existing work in DRL is focused on learning factors which are uncorrelated. In our setting, however, the different facets are often highly correlated (e.g. natural disaster movies typically have a high budget).

In this paper, we explore a strategy for decomposing a given vector space embedding into separate facet spaces by first determining which interpretable features are modelled by the vector space and then clustering the word vectors corresponding to these features. Despite being intuitive, given that word embeddings are known to group together functionally similar words, we found this strategy to perform poorly in its basic form. First, simply looking for clusters in word embeddings often leads to thematic clusters, e.g. grouping *horror* together with words such as *scary* and *zombie* rather than other genres such as *western* and *drama*. To address this, we explicitly prevent two words from ending up in the same cluster if the features they are modelling are too similar. Second, in most domains, there are one or two central facets which tend to be highly correlated with most of the other facets (e.g. genre in the movie domain). To ensure that the resulting facet spaces are sufficiently different (rather than capturing minor variations of the most central facets), we found it useful to use an iterative approach, where previously found facets are "removed" from the vector space embedding before proceeding to find further facets. With these two modifications, we find that useful facets can indeed be found, which consistently lead to better classification performance compared to the original vector space embedding.

2 Related Work

Conceptual Spaces. A conceptual space (Gärdenfors, 2000) is a vector representation of the entities from some domain, where the dimensions tend to capture salient features. It is usually assumed that the dimensions of a conceptual space can be grouped into semantic domains, or facets. From a cognitive point of view, this grouping is important because it affects how similarity scores are computed. Intuitively, this is because the dimensions from the same facet tend to interact with each other whereas the dimensions from different facets can be considered in isolation. The problem of learning conceptual spaces from data has only received limited attention to date. One exception is the work of Derrac and Schockaert (2015), which we build on in this paper. In their work, textual descriptions of the considered entities are used to find dimensions that model salient semantic features in a given semantic space. For instance, in a semantic space of movies they found dimensions corresponding to features such as *scary*, *horror* and *zombie*. Note that because these features tend to be correlated, the corresponding dimensions are typically not orthogonal in the input semantic space. For this reason, they refer to these dimensions as *interpretable directions*. More recently, (Ager et al., 2018) proposed a post-processing method to fine-tune these interpretable directions. The main challenge which we address in this paper is to group the features that are found by the method from Derrac and Schockaert (2015) into semantically meaningful facets. A supervised variant of this problem was considered by Banaee et al. (2018). Their approach relies on feature selection methods to find subsets of features that are predictive of particular class labels, based on a set of labelled training examples. In contrast, our focus in this paper is on unsupervised methods, as suitable training data is often not available.

Disentangled Representation Learning (DRL). In the last few years, a large number of generative neural network models have been proposed, with variational autoencoders (VAEs) (Kingma and Welling, 2014) and generative adversarial networks (GANs) (Goodfellow et al., 2014) being the best-known examples. The main underlying idea behind these models is that high-dimensional data (e.g. images) can often be described in terms of a much lower-dimensional latent vector space. Each object can thus be compactly described by its latent code, i.e. the corresponding vector in this latent space. The problem of DRL is to learn such a latent vector representation which is such that (groups of) the dimensions of the latent codes correspond to meaningful interpretable factors. A variety of unsupervised and semi-supervised approaches for learning such disentangled representations have been proposed, such as InfoGAN (Chen et al., 2016), which is based on a modification of the loss function for GANs, and β-VAE (Higgins et al., 2017), which instead uses VAEs as the base model. Conceptually, these approaches modify the loss function of a given generative

model by insisting that the dimensions of the latent vector space are in some sense independent. In principle, the latent vector spaces learned by DRL methods can be viewed as conceptual spaces. It is unclear, however, whether purely statistical measures of independence can be sufficient for learning semantically meaningful factors. While interesting results have been obtained for particular applications, after a thorough empirical analysis Locatello et al. (2018) concluded that such results were highly sensitive to the random initialization of the neural network models and the value of hyper-parameters. Their results suggest that, in absence of a suitable supervision signal, high-quality factors can only be learned in the presence of a strong inductive bias. Going beyond unsupervised approaches, (Jain et al., 2018) propose a supervised approach for DRL for text. As supervision signal, they use triplets of the form $(s, d, o)_a$ which encode that relative to aspect a, it holds that s and d are more similar than d and o. Then they use a Convolutional Neural Network (CNN) based model to obtain low-dimensional document embeddings for each considered aspect.

3 Decomposing Conceptual Spaces

Let E be a set of entities of some particular type (e.g. movies) for which a vector space embedding is given. In the following, we will write $\mathbf{e} \in \mathbb{R}^n$ for the embedding of entity e. The first step of our approach consists in applying the method from Derrac and Schockaert (2015), which provides us with a set of words F, each corresponding to a feature that can be modelled as a direction in the vector space. For $f \in F$ we write $\mathbf{d_f}$ for the vector characterizing this direction. Formally, this means that $\mathbf{e_1} \cdot \mathbf{d_f} < \mathbf{e_2} \cdot \mathbf{d_f}$ iff e_2 has the feature f to a higher extent than e_1 (e.g. if f denotes the feature *scary*, then this would mean that movie e_2 is scarier than movie e_1). We briefly recall the method from Derrac and Schockaert (2015) in Section 3.1. Our hypothesis is that we can group these features into meaningful facets and that we can represent these facets as subspaces of the given vector space embedding. Section 3.2 discusses our approach for finding these subspaces.

3.1 Identifying Feature Directions

The method proposed in Derrac and Schockaert (2015) aims to finds a set of features F which can be modelled as directions in the given vector space. The input to their method consists of a text description D_e of each entity e, but they assume no other prior knowledge. In particular, each word w which occurs sufficiently frequently in the document collection $\mathcal{D} = \{D_e \,|\, e \in E\}$ is considered as a candidate feature. To determine whether w should be added as a feature, they train a linear SVM classifier to separate the vector representations of the entities e for which w is mentioned in D_e from the vector representations of the other entities. If this SVM classifier is sufficiently accurate[2], they assume that the word w captures a salient feature. The corresponding feature direction is then characterized by the normal vector $\mathbf{d_f}$ of the hyperplane that was learned by the SVM classifier. We will use the notation pos_w to refer to the set of entities from E which are classified as positive. In our experiments, we used logistic regression classifiers instead of SVMs, which we found to perform similarly but were faster to train.

3.2 Finding Facets

Our aim is to group the features from F into meaningful facets. For instance, in the movies domain, we might expect to see facets corresponding to e.g. genre, language and release date. It does not seem possible (nor desirable) to formally define what constitutes a good facet, a typical problem in unsupervised learning. Intuitively, however, a facet should group features which are of the same kind (e.g. genres) and should in some sense be exhaustive (i.e. all genres, rather than a set of features that refer to one or a few particular genres).

Using subspace clustering. The aim of subspace clustering is to decompose a high-dimensional space into the union of lower-dimensional spaces. This problem has found numerous applications, especially in computer vision. One may wonder whether we can learn useful facets by applying subspace clustering to feature directions $\mathbf{d_f}$. Unfortunately, in our initial experiments, this approach did not prove successful. This is illustrated for the movies domain in Table 1, where we used the state-of-the-art SSC-OMP subspace clustering method (You et al., 2016). For this comparison, we first manually grouped the features from F to obtain a gold standard. The first column of the table shows two of the resulting facets: one corresponding to genres and one corresponding to different

[2]Because of class imbalance, they used Cohen's Kappa instead of classification accuracy.

MOVIES		
Gold standard	**IncHDB**	**SSC-OMP**
blu, ray, cgi, dolby, surround, computer, technology, theaters, theatre, purchased, ordered, purchase, dvds, amazon, bought, copy, audio, disc, edition, widescreen, transfer, digital, print, vhs, discs	**Initial cluster X_i:** audio, disc, dvds, digital, vhs, dolby, technology, discs, computer, version **Top additional features Y_i:** transfer, edition, blu, cgi, ray, widescreen, amazon, extras, awesome, computer, purchased, purchase, buying, surround, price, trailer, included, favorites, theaters, alot, previews, extra, player	blu, arts, disc, purchased, edition, ordered, crime, british, creepy, disturbing, fighting, charm, rent, honestly, reviewers, oscar, personally, excited, questions, budget, england, education, victims, packed, marriage, tense, detail, fell, hell, deeply, culture, situation, accurate, trailers
thriller, comedic, comedy, documentary, comic, satire, documentaries, drama, melodrama, horror,action, adults, animation, crime, fantasy,family, musical, mystery, romance, war, western	**Initial cluster X_i:** thriller, comedic, comedy, documentary, comic, satire, documentaries, humor, humour, cheesy, adaptation, wit, melodrama, campy, parody **Top additional features Y_i:** hilarious, gags, laughs, jokes, slapstick, funniest, thrillers, funnier, suspense, witty, unfunny, amusing, suspenseful, historical, horror, romance, interviews, psychological	horror, thriller, political, charming, funnier, slapstick, documentaries, hilarious, killed, seat, issue, cheesy, gory, mystery, effects, amazon, widescreen, transfer, realistic, relationship, monster, epic, portrayed, glad, premise, hearing, evil, car, formula, decision, violent, villain, gun, goofy, game, teens, garbage, humor, ruin, product, amount, dad, loving, personality, award, folks

Table 1: Comparison of learned facets with gold standard for the movies domain.

media types (which indirectly captures the time period during which a movie was released). The right-most column shows the closest facets that were found with SSC-OMP. As can be seen, these facets are largely non-sensical. For instance, in the first case, words such as *blu* and *disc* are clustered together with semantically unrelated words such as *fighting*, *england* and *accurate*. In the second example, genres such as *horror* and *thriller* are grouped together with unrelated words such as *cheesy*, *widescreen* and *award*. This negative result seems in accordance to the findings from Locatello et al. (2018) that unsupervised disentangled representation learning seems impossible without a strong inductive bias. We also tried several other subspace clustering methods, for a wide range of different configurations, without obtaining better results. Similarly, we experimented with neural approaches for learning disentangled representations directly from the bag-of-words representations of the entities, but again unsuccessfully.

Using word embeddings. These negative results strongly suggest that some kind of external knowledge is needed to find meaningful facets. To this end, we focus on the use of word embeddings, which seems natural given the fact that words of the same kind (e.g. different names of genres) tend to be used in similar contexts, and can thus be expected to have similar word vectors. In particular, our basic approach for identifying facets consists in clustering the word vectors, from some standard pre-trained word embedding model, corresponding to the features in F. One important drawback of this basic strategy, however, is that it often leads to thematic clusters. For instance, while we would want *horror* to be clustered together with other names of genres, when simply clustering word vectors without any further guidance, *horror* may be clustered together with thematically similar words such as *scary* and *zombie*. To avoid such clusters, we rely on the insight that if a and b are thematically similar words (e.g. *horror* and *zombie*) then the corresponding feature directions $\mathbf{d_a}$ and $\mathbf{d_b}$ will also be similar. However, for paradigmatically similar words, such as *horror* and *comedy*, this should not be the case. In other words, two words should intuitively end up in the same clusters if they have similar word vectors but dissimilar feature directions.

While there are many ways to implement this intuition, we found that using the cosine similarity between $\mathbf{d_a}$ and $\mathbf{d_b}$ was not always reliable. Instead we rely on the following measure of overlap between the sets pos_a and pos_b:

$$o(a,b) = \min\left(\frac{|pos_a \cap pos_b|}{|pos_a|}, \frac{|pos_a \cap pos_b|}{|pos_b|}\right)$$

The dissimilarity between features a and b from F is then defined as follows:

$$d(a,b) = \begin{cases} 1 - \cos(\mathbf{w}_a, \mathbf{w}_b) & \text{if } o(a,b) \leq \lambda \\ 1 & \text{otherwise} \end{cases}$$

where the overlap threshold λ is a hyperparameter and $\mathbf{w}_f$ denotes the word vector for feature f.

The aim of the clustering step is to find a number of disjoint subsets of F, each of which intuitively corresponds to a facet. We will denote these facets by $X_1, ..., X_k$. To avoid finding redundant facets, we identify them in an incremental fashion. In particular, from the clusters obtained by the clustering algorithm, we only select the single most important one, i.e. the one which is most

likely to describe a salient facet. For this purpose, we rank clusters according to the following score:

$$score(X_i) = |\bigcup_{f \in X_i} pos_f| \quad (1)$$

This score reflects the intuition that we prefer clusters with features that are general and diverse, i.e. such that most of the entities would have at least one of the features from the cluster. As will be explained below, after the subspace corresponding to this facet has been determined, we iteratively apply the same method on a reduced vector space to find the next most important facet, until the desired number of facets k has been found.

3.3 Modelling Facets as Subspaces

We model each facet X_i as a linear subspace of the given vector space embedding. To find this subspace, we learn new feature directions $\mathbf{c_f}$ for each $f \in X_i$, which still capture these features but lie in a low-dimensional subspace. In particular, we minimize the following objective:

$$\sum_{e \in E} \sum_{f \in X_i} \log \sigma(\mathbf{c_f e} + b_f) \quad (2)$$

where

$$\mathbf{c_f} = \lambda_1^f \mathbf{a_1^i} + \lambda_2^f \mathbf{a_2^i} + ... + \lambda_r^f \mathbf{a_r^i} \quad (3)$$

with r the desired number of dimensions of the subspace. Note that (2) essentially expresses that for each $f \in X_i$, we want to train a logistic regression classifier with coefficient vector $\mathbf{c_f}$. However, as expressed in (3), rather than learning these coefficient vectors independently, they are constrained such that they can be written as a linear combination of the vectors $\mathbf{a_1^i}, ..., \mathbf{a_r^i}$. The resulting feature directions thus span a subspace of (at most) r dimensions. Let $\mathbf{M}_i \in \mathbb{R}^{r \times n}$ be an orthonormal basis for this subspace. Then $\mathbf{e}^i = \mathbf{M}_i \mathbf{e}$ is the r-dimensional facet-specific embedding of entity e.

There may be some features from F which are not contained in X_i but can nonetheless be modelled well in the resulting subspace (i.e. if they are semantically related to the features in X_i). To identify these features, we apply the method from (Derrac and Schockaert, 2015) to the facet-specific embeddings. We write Y_i to denote the features that were thus identified, beyond the ones from X_i.

Next we determine a null space of the basis $\mathbf{M}_i$, i.e. an $(n-r) \times n$ dimensional matrix $\mathbf{R}_i$ satisfying

$$\mathbf{M}_i \mathbf{R}_i^T = \mathbf{0}$$

Dataset	Entities	Attribute	Classes
Movies	13978	Keywords	100
		Genre	23
		Ratings	6
place-types	1383	Foursquare (Fours.)	9
		Geonames (Geo.)	7
		OpenCYC (OpenC.)	20
Organisation	11800	Country	4
		Headquarter Location(HL)	2
Building	3721	Country	2
		Administrative Location(AL)	2

Table 2: Overview of considered datasets.

This matrix $\mathbf{R}_i$ is a basis for the orthogonal complement of the subspace spanned by $\mathbf{M}_i$. Intuitively, it defines what remains of the vector space embedding after we remove (i.e. project away) the subspace modelling the facet X_i.

To find the remaining facets, we repeat the same procedure, but with two changes. First, the $n - r$ dimensional remainder space is used instead of the original embedding space, i.e. we use $\mathbf{R}_i \mathbf{e}$ as the vector representation of e. Second, the features in $X_i \cup Y_i$ are no longer considered by the clustering algorithm. This process is repeated until the desired number of facets has been found, each time considering an increasingly lower-dimensional remainder space and clustering only those features that are not already modelled in a previously identified facet. Intuitively, by learning the facets in this incremental way, we should be able to avoid finding multiple variants of the same facets.

The middle column of Table 1 shows two of the facets that were found with this approach. Intuitively, these facets are clearly more meaningful than those that were found with SSC-OMP.

4 Experimental Analysis

Methods. We have experimented with two clustering algorithms: agglomerative hierarchical average link clustering and HDBSCAN (Campello et al., 2013). However, in the case of HDBSCAN we noticed that when using overlap-based dissimilarity, we typically ended up without any clusters[3]. For HDBSCAN we therefore used cosine similarity instead. We refer to our method with agglomerative clustering as *IncAgg* and to the variant with HDBSCAN as *IncHDB*. In addition, we considered a variant of the method with agglomerative clustering which relies on cosine similarity instead of the overlap-based dissimilarity (*CosIncAgg*).

[3]Note that HDBSCAN does not cluster all the data points, as it removes data points which are considered as noise.

		Place types			Movies			Organisations		Buildings	
		Fours.	Geo.	OpenC.	KeyW.	Genre	Rating	Country	HL.	Country	AL.
DT-D1	MDS	0.34	0.26	0.26	**0.26**	0.38	0.43	0.67	0.24	0.47	0.47
	IncAgg	**0.45**	**0.30**	**0.30**	0.25	**0.40**	**0.47**	**0.76**	**0.26**	**0.50**	**0.50**
	CosIncAgg	0.45	0.26	**0.30**	0.24	0.38	0.43	0.75	0.23	0.43	0.42
	IncHDB	0.43	0.26	0.28	0.25	0.38	0.40	0.50	0.22	0.46	0.46
	NonIncHDB	0.30	0.20	0.27	0.23	0.34	0.40	0.50	0.20	0.46	0.47
	NonIncAgg	0.33	0.24	0.27	0.23	0.33	0.42	0.40	0.21	0.48	0.47
DT-D3	MDS	0.52	0.27	0.32	**0.27**	**0.43**	0.47	0.70	0.27	0.47	0.46
	IncAgg	**0.58**	**0.34**	**0.34**	**0.27**	0.41	**0.47**	0.77	**0.30**	**0.54**	**0.52**
	CosIncAgg	0.54	0.28	**0.34**	0.25	0.40	0.45	**0.78**	0.26	0.47	0.45
	IncHDB	0.57	0.26	0.31	**0.27**	0.41	0.45	0.70	0.27	0.49	0.50
	NonIncHDB	0.43	0.24	0.27	0.26	0.38	0.44	0.60	0.21	0.48	0.49
	NonIncAgg	0.36	0.30	0.29	0.24	0.38	0.45	0.65	0.22	0.51	0.50
SVM	MDS	0.65	0.31	0.35	0.25	**0.54**	0.54	0.71	**0.26**	0.38	0.39
	IncAgg	**0.73**	0.33	**0.37**	**0.26**	**0.54**	**0.55**	0.76	**0.26**	**0.52**	**0.51**
	CosIncAgg	0.62	0.33	0.34	0.25	0.52	0.53	**0.80**	0.12	0.50	0.50
	IncHDB	0.65	0.30	0.36	0.23	0.50	0.51	0.70	0.20	0.51	**0.51**
	NonIncHDB	0.60	**0.35**	**0.37**	0.24	0.46	0.52	0.68	0.24	**0.52**	**0.51**
	NonIncAgg	0.58	**0.35**	0.35	0.24	0.48	0.51	0.72	0.26	0.50	**0.51**
Gaussian	MDS	0.81	0.45	**0.46**	0.26	0.58	0.48	0.74	0.27	0.53	0.51
	IncAgg	**0.87**	**0.48**	0.45	**0.28**	**0.60**	**0.51**	**0.81**	0.27	0.54	**0.55**
	CosIncAgg	0.81	0.45	**0.46**	**0.28**	**0.60**	**0.51**	**0.81**	**0.28**	0.53	0.53
	IncHDB	0.84	0.43	0.43	0.27	**0.60**	**0.51**	0.80	**0.28**	0.54	0.53
	NonIncHDB	0.75	0.41	0.40	0.23	0.51	0.47	0.75	0.27	**0.59**	0.53
	NonIncAgg	0.71	0.46	0.45	0.22	0.52	0.46	0.77	0.27	0.58	0.53

Table 3: Classification tasks performance (in terms of F1 score) when using the MDS space and four variation of the facet-based representations.

Finally, we also report results for variants of our methods in which we did not obtain the facets incrementally (*NonIncAgg* and *NonIncHDB*). In these cases, we simply extract r clusters from the initial set of features F and determine the corresponding facets directly. In all cases, we use 50-dimensional pre-trained GloVe word vectors (Pennington et al., 2014) for clustering the features.

To generate the initial vector space embedding, we follow the approach proposed in (Derrac and Schockaert, 2015) based on multi-dimensional scaling. In all cases, we used 100-dimensional vector spaces and learned 10 facets, each being modelled as a 10-dimensional subspace. To select the set of features F, we initially consider the 500 highest scoring words according to the Kappa metric. However, if we end up without any clusters (in the case of HDBSCAN), we expand the set of features to the 1000 top words. The overlap threshold λ is selected based on held-out tuning data, considering values from $\{0.3, 0.5, 0.7\}$. To flatten the agglomerative clustering, we tune the number of clusters from $\{50, 100, 200\}$[4].

[4]The source code is available online at https://github.com/rana-alshaikh/Disentangled-Facets.

Evaluation tasks. Intrinsic evaluation of the learned facets is difficult, among others because what we might consider to be a natural facet is highly subjective. Therefore, in our quantitative evaluation, we will focus on the impact of the learned facets in a number of classification tasks. This is also motivated by the view that some types of classifiers need semantically meaningful features to perform well. For example, Ager et al. (2018) used low-depth decision trees to evaluate a method for learning feature directions in vector space embeddings. Specifically, if $F = \{f_1, ..., f_m\}$ is the set of features that were identified, then they represent each entity e using the feature vector $(\mathbf{d_{f_1}} \cdot \mathbf{e}, ..., \mathbf{d_{f_m}} \cdot \mathbf{e})$, with $\mathbf{d_f}$ the direction modelling feature f as before. Given that a depth-1 decision tree can only use one of these features, the performance of such a decision tree essentially tells us to what extent the classes that are considered in the supervised classification task have been discovered as features. In our experiments, we will report the result of depth-1 and depth-3 decision trees. As the baseline method, referred to as *MDS*, we will use the top-2000 features that we obtained with the method from Derrac and Schockaert (2015). To evaluate the facets,

Organisations	
Cosine similarity	**Overlap-based dissimilarity**
union, europe, protection, aid, spain, right, cross, war, players, black, defenders, german, european	canadian, australian, australia, africa, nations, african, canada, states, countries, asia, united, british, european, competition, world, europe, asian, britain, country, german
Place types	
landscapes, serene, tranquil, closeup, surreal, greenery, scenery, scenic, breathtaking, picturesque	sculptures, decoration, churches, fauna, historical, landscapes, st, archeology, small, sculpture, basilica, monuments, convent, heritage, artistic, monument, sacred, forgotten, cemetery, baroque, festival, promenade, renaissance, hall, flora, pavilion, memorial

Table 4: Examples of clusters when standard cosine similarity is used (left) and with the proposed overlap based dissimilarity score (right).

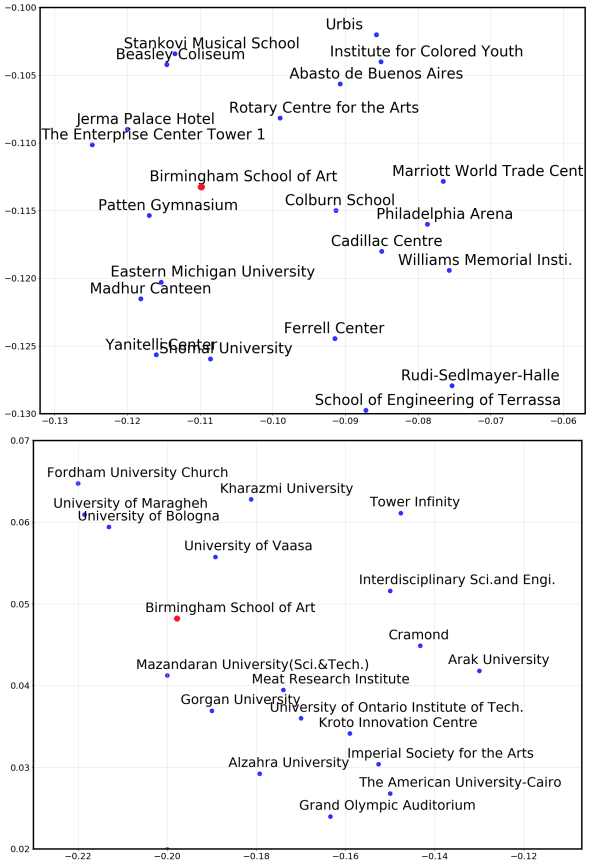


Figure 1: Projection of a 100-dimensional semantic space and 10-dimensional facets of buildings. Top: showing the full space. Bottom: showing the 10-dimensional representations for the facet $X_i = \{$ campuses, students, offices, centers, facilities, area, hotels, homes, bridges, hospitals, cities, shops, stations$\}$

we instead apply this method to find the top-200 features for each of the facet subspaces.

The performance of the decision trees will allow us to evaluate whether we are able to learn higher-quality feature directions thanks to the decomposition of the vector space into facet subspaces. To evaluate the quality of the facets independently of the quality of the feature directions, we also consider classifiers which use as input the facet-specific vector representations $\mathbf{e}^i$ of the entities. Specifically, we train a support vector machine (SVM) for each of the facets, leading to the predictions $p_1, ...p_k$. These predictions are then aggregated to a final prediction using a logistic regression meta-classifier. As baseline, we simply train a single SVM classifier in the full vector space. As our final classifier, we estimate a Gaussian model from the positive training examples. In particular, we estimate a univariate Gaussian for each dimension and multiply the corresponding probabilities. We chose this method because it is sensitive to how well the dimensions of the space are aligned with semantically meaningful properties, and because such Gaussians are commonly used for representing categories in conceptual spaces (Bouraoui and Schockaert, 2018). For the baseline, we use the dimensions of the full vector space. For the facet-based representations, we use the dimensions of the facet subspaces.

Dataset. We have carried out experiments with vector space embeddings for four different domains. First, we used the *movies* and *place type* domains from (Derrac and Schockaert, 2015), where the embeddings are learned respectively from movie reviews and from Flickr tag co-occurrence distributions. We also considered two additional domains, for which we used Wikipedia articles: *buildings* and *organisations*. In particular, we retrieved all Wikipedia pages whose semantic type on WikiData corresponds to building or organisation. Wikipedia pages containing fewer than 200 words were removed. The bag-of-words (BoW) representation of the remaining Wikipedia concepts were obtained using a standard preprocessing strategy (e.g. removing HTML tags and references), including stopword removal with NLTK (Bird and Loper, 2004). Furthermore, we POS tagged the documents and only retained the nouns and adjectives. Finally, frequent words that occurred in more than 60% of the Wikipedia articles about buildings (resp. organisations) were removed, as well as words that occurred fewer than 10 times. This approach was taken to stay broadly in line with the strategy that was used in (Derrac and Schockaert, 2015). As classification tasks, we used two attributes from WikiData in both domains (being the only attributes for which

BUILDINGS	
Initial cluster X_i:	**Top additional features Y_i:**
architecture, art, history, literature, architectural, society, culture, ancient, scholars, vernacular, classical, historical, contemporary, cultural, medieval	structure, floors, county, architect, graduate, palace, revival, property, hall, united, floor, farm, design, art, space, style, states, downtown, interior, mansion, arena, architectural, architecture, chemistry, entrance.
city, district, town, rural, central, cities, neighborhood, areas, part, section, province, village, north, western, portion, middle, residents, between, branch	company, wife, headquarters, firm, area, events, estate, people, facility, streets, avenue, schools, hotel, commercial, former, states, architects, original, farm, example, residence,

Table 5: Examples of the facets from the Buildings dataset (using *IncAgg* method).

a sufficient number of entities per attribute value was found). The full datasets will be released upon acceptance. The properties of the considered domains and associated classification problems are summarized in Table 2. For each classification problem, we randomly split the labelled examples into 2/3 for training and 1/3 for testing. For tuning we use 5-fold cross-validation over the training set. In the movies domain, where more labelled data is available, we have used fixed splits of 60% for training, 20% for tuning and 20% testing.

Results. The results are summarized in Table 3. Our main method *IncAgg* outperforms the *MDS* baseline for almost all classification tasks and types of classifiers. For the HDBSCAN based variant, the results are more mixed, which seems related to the fact that the overlap based dissimilarity could not be used in that case. Indeed, the cosine based variant of *IncAgg*, i.e. *CosIncAgg*, also performs consistently worse than *IncAgg*. Looking at the performance of *NonIncAgg* and *NonIncHDB* reveals that learning facets in an iterative fashion is critical, given that these two variants perform worse than the baseline in many cases.

Looking more closely at the results of our main method *IncAgg*, it is interesting to note that large improvements are obtained for depth-1 decision trees, which shows that our facet subspaces make it easier to identify features that correspond to the categories from the corresponding classification problems. However, large improvements can also be seen for SVMs, which shows that the actual decomposition of the space is also helpful.

Qualitative Analysis. Figure 1 illustrates how our subspaces capture similarity in a facet-specific way, showing the two first principal components of the embedding of *Birmingham School of Art* in the full space and in the subspace of a facet that intuitively captures building type. While the neighbours in the full space are a mixture of different building types (hotels, commercial buildings, museum, and educational buildings), in the facet subspace all nearest neighbors are universities.

Table 4 illustrates the impact of using overlap-based dissimilarity, where the clusters obtained with cosine similarity are clearly more thematic, while the ones obtained with the overlap-based metric intuitively capture a facet (i.e. geographic location and the natural–cultural opposition). Finally, Table 5 shows some of the facets obtained in the buildings domains. The first example shows a facet which intuitively captures the historical–contemporary opposition, while the second example shows a facet that captures the rural–city opposition.

5 Conclusions

We considered the problem of decomposing a vector space embedding into facets, which are characterized by a set of semantically related features and a corresponding subspace of the embedding. In particular, we focused on unsupervised methods, considering both approaches that rely on the vector space itself (i.e. using subspace clustering) and approaches that additionally take into account the information about word meaning that is captured by pre-trained word vectors. Overall, we found this problem to be highly challenging, in accordance with the findings from Locatello et al. (2018) regarding unsupervised disentangled representation learning. However, we were still able to obtain useful facets based on two crucial modifications to a standard clustering based strategy. First, we measure the similarity between features based on two factors: the similarity between their word vectors and the dissimilarity between their meaning in the vector space embedding (measured in terms of overlap). Second, we found it essential to learn facets in an iterative fashion, to avoid too much redundancy between the different facets.

Acknowledgments

Steven Schockaert was supported by ERC Starting Grant 637277. Zied Bouraoui was supported by CNRS PEPS INS2I MODERN.

References

Thomas Ager, Ondrej Kuzelka, and Steven Schockaert. 2018. Modelling salient features as directions in fine-tuned semantic spaces. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 530–540.

Hadi Banaee, Erik Schaffernicht, and Amy Loutfi. 2018. Data-driven conceptual spaces: Creating semantic representations for linguistic descriptions of numerical data. *Journal of Artificial Intelligence Research*, 63:691–742.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Zied Bouraoui and Steven Schockaert. 2018. Learning conceptual space representations of interrelated concepts. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 1760–1766.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 160–172.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Joaquín Derrac and Steven Schockaert. 2015. Inducing semantic relations from conceptual spaces: a data-driven approach to plausible reasoning. *Artificial Intelligence*, 228:66–94.

Igor Douven, Lieven Decock, Richard Dietz, and Paul Égré. 2013. Vagueness: A conceptual spaces approach. *Journal of Philosophical Logic*, 42:137–160.

P. Gärdenfors. 2000. *Conceptual Spaces: The Geometry of Thought*. MIT Press.

Peter Gärdenfors. 1996. Mental representation, conceptual spaces and metaphors. *Synthese*, 106:21–47.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. β-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Sarthak Jain, Edward Banner, Jan-Willem van de Meent, Iain J Marshall, and Byron C Wallace. 2018. Learning disentangled representations of texts with application to biomedical abstracts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4683–4693.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations*.

Francesco Locatello, Stefan Bauer, Mario Lucic, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. 2018. Challenging common assumptions in the unsupervised learning of disentangled representations. *CoRR*, abs/1811.12359.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Peter D Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Chong You, Daniel Robinson, and René Vidal. 2016. Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3918–3927.

Weird Inflects but OK: Making Sense of Morphological Generation Errors

Kyle Gorman[*], **Arya D. McCarthy**[†], **Ryan Cotterell**[†],
Ekaterina Vylomova[‡], **Miikka Silfverberg**[§], and **Magdalena Markowska**[*]
[*]The Graduate Center, City University of New York, [†]Johns Hopkins University,
[‡]University of Melbourne, [§]University of Helsinki
kgorman@gc.cuny.edu, arya@jhu.edu, ryan.cotterell@jhu.edu,
evylomova@gmail.com, miikka.silfverberg@helsinki.fi,
mmarkowska@gradcenter.cuny.edu

Abstract

We conduct a manual error analysis of the CoNLL–SIGMORPHON 2017 Shared Task on Morphological Reinflection. In this task, systems are given a word in citation form (e.g., *hug*) and asked to produce the corresponding inflected form (e.g., the simple past *hugged*). This design lets us analyze errors much like we might analyze children's production errors. We propose an error taxonomy and use it to annotate errors made by the top two systems across twelve languages. Many of the observed errors are related to inflectional patterns sensitive to inherent linguistic properties such as animacy or affect; many others are failures to predict truly unpredictable inflectional behaviors. We also find nearly one quarter of the residual "errors" reflect errors in the gold data.

1 Introduction

A huge amount of work in natural language processing treats words as indivisible units, but the vast majority of the world's languages have rich word-internal structure. For instance, 80% of the languages analyzed in the *World Atlas of Linguistic Structure* (Dryer and Haspelmath 2013) inflect verbs for tense and 65% inflect nouns for case. Generating and processing complex words is thus crucial for multilingual speech and language technologies.

Recent work on large-scale, multilingual computational modeling of morphology (e.g., Durrett and DeNero 2013, Cotterell et al. 2016) targets supervised **inflection generation**. Such tasks require variable-length outputs, so they are less constrained than earlier segmentation-based tasks (e.g., Kurimo et al. 2010), but appear to be tractable with existing neural network–based models. For example, in the CoNLL–SIGMORPHON 2017 Shared Task (sub-task 1 and the "high-data condition"), the focus of this study, the best-ranked system generates novel inflectional forms with 90% accuracy or better for 46 out of the 52 target languages, It achieves perfect accuracy for four languages (Cotterell et al. 2017).

In light of these apparent success, we examine the failure modes of existing models for morphological generation. We first propose and motivate an error taxonomy for this task, inspired by similar proposals for other natural language generation and processing technologies such as grammatical error correction (e.g., Rozovskaya and Roth 2016) and machine translation (e.g., Popović and Ney 2011, Fishel et al. 2012, Irvine et al. 2013). We then use this taxonomy to perform a manual error analysis of the CoNLL–SIGMORPHON 2017 Shared Task. Such analyses can help to identify strengths and weaknesses of existing systems, suggest future improvements, and guide development of strong ensemble models, but are often neglected or treated as an afterthought. This annotation also allows us to measure the quality of the gold data.

Generating morphologically complex forms is a skill typically-developing children effortlessly acquire, so this task, and systems' error patterns, may have implications for the theory of language acquisition. While the shared task training paradigm is quite unlike human language learning, inference and evaluation resemble the classic *wug*-test (Berko 1958), in which speakers are presented with a word—either real or nonce—in citation form and prompted to provide a particular inflectional form of that word. Therefore, one can analyze inflection generation errors much like how one might analyze errors made by a child acquiring their first language. And, one can ask whether humans' and artificial learners' errors are in any way alike.

To answer these questions, we examine errors made by the two top-performing systems in the CoNLL–SIGMORPHON 2017 Shared Task for twelve languages.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 140–151
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

2 Materials and methods

Here, we describe the shared task, data sources, and the targeted systems.

2.1 The task

The CoNLL–SIGMORPHON 2017 Shared Task (Cotterell et al. 2017) consists of two supervised morphological generation sub-tasks across 52 languages. In sub-task 1, the training data consists of triples of lemma, inflectional bundle, and inflected form, as in Table 1. At inference time, the system is given lemmata and inflectional bundles and asked to produce the appropriate inflected forms. In sub-task 2, training data consists of complete inflectional paradigms, and at inference time, the system is asked to produce full paradigms for unseen lemmata. We focus on the results from sub-task 1, primarily because only two of the twelve teams chose to compete in sub-task 2. However, the proposed error taxonomy could easily be applied to sub-task 2, or to later morphological generation challenges such as sub-task 2 of the CoNLL–SIGMORPHON 2018 shared task (Cotterell et al. 2018) or sub-task 1 of the SIGMORPHON 2019 shared task (McCarthy et al. 2019).

2.1.1 Data

The data in both sub-tasks is primarily sampled from UniMorph (Kirov et al. 2016, 2018), a free morphological database. In turn, UniMorph data for our twelve languages is automatically extracted from Wiktionary, a collaborative multilingual online dictionary. UniMorph pairs the cells of Wiktionary morphological paradigms, which bear prose labels like "genitive plural", to feature bundles in a language-independent morphological schema (Sylak-Glassman et al. 2015; also see Sylak-Glassman 2016). The data consist of the aforementioned triples of lemma, inflectional bundle, and inflected form. For sub-task 1, these triples were sampled from UniMorph paradigms according to frequencies of inflected forms as estimated from Wikipedia. Because of this sampling procedure, the data is sparse in the sense that there are rarely more than a few inflected forms per lemma. As such, this roughly mimics the statistical properties of the primary linguistic data encountered by child language learners (e.g., Chan 2008:71–100). Systems were evaluated under three training data conditions: low (100 triples), medium (1,000 triples) and high (10,000 triples). We focus on the high-data condition because nearly all systems performed poorly in the low- and medium-data conditions.

2.2 Systems

In sub-task 1, systems were ranked using the macro-averaged "per form" (i.e., full-token match) accuracy across all 52 target languages.[1] We analyze errors made by the two top-ranked systems, briefly described below.

UE-LMU-I **(Bergmanis et al. 2017)** This system uses a recurrent neural network (RNN) with a bidirectional gated recurrent unit (GRU) encoder, a unidirectional GRU decoder, and a standard attention mechanism. It enhances a closely-related competitor system (Kann and Schütze 2017) by augmenting the provided training data with identical input-output pairs so as to create a bias toward copying the input stem. It is ranked as the best-performing system on sub-task 1 (macro-average accuracy 95.32%).

CLUZH-7 **(Makarov et al. 2017)** This system also uses a neural encoder-decoder but replaces the "soft" attention mechanism with hard monotonic attention (Aharoni and Goldberg 2017) and special edit operations. It is ranked second-best overall on sub-task 2 (macro-average accuracy 95.12%) and also achieves the highest per form accuracy on eight languages including Hungarian and Spanish.

3 Error taxonomy

One major distinction in the proposed taxonomy of inflection generation errors is between those errors which can be given a linguistic characterization—i.e., in terms of misapplication of inflectional patterns independently attested in the target language—from those which cannot. As such we are inspired by a long and contentious debate in computational morphology research. Rumelhart and McClelland (1986) propose an early neural network model trained to generate the phonological form of English simple past tense verbs given the present tense. They claim that under in certain conditions, their model produces errors that are similar to those made by children acquiring English, such as **catched* for *caught*.[2]

[1] The other metric used in sub-task 1, average Levenshtein distance between word and target averaged over languages, ranks systems nearly identically (Cotterell et al. 2017:11).

[2] Such errors are known as **overregularizations** in the language acquisition literature (e.g., Marcus et al. 1992).

Language	Lemma	Inflection	Inflected form
English	hug	V;PST	hugged
	spark	V;V.PTCP;PRS	sparking
German	aufbauen	V;IND;PRS;2;SG	baust auf
	Ärtzin	N;DAT;PL	Ärtzinnen
Spanish	descomponer	V;NEG;IMP;2;PL	no descompagáis
	liberar	V;IND;FUT;2;SG	liberarás

Table 1: Sample training data for sub-task of the CoNLL–SIGMORPHON 2017 Shared Task. Each training example maps a **lemma** (a citation form) and **inflection** (a bundle of UniMorph morphosyntactic features) to an **inflected form**. At inference time, the inflected form is predicted given a lemma and inflection.

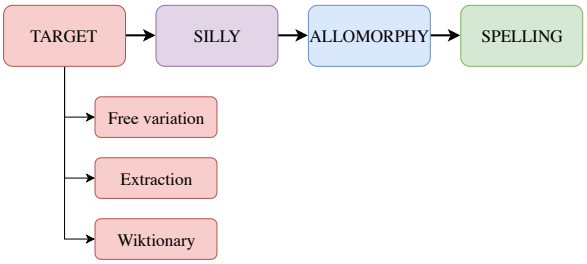


Figure 1: Overview of our annotation scheme, including subcategories. Annotators are instructed to proceed through the taxonomy from left to right.

Pinker and Prince (1988) and Sproat (1992:216f.) dispute this characterization, pointing out bizarre errors like **membled* for *mailed*. More recently, Kirov and Cotterell (2018) claim that modern neural network architectures—such as those used in the CoNLL–SIGMORPHON 2017 Shared Task—generalize reasonably well while largely eliminating these bizarre errors. However, Corkery et al. (2019) argue that the Kirov and Cotterell model predictions align poorly with human productions, and suggest that the reported results may be uncharacteristic due to fortuitous random seeding.

We desired a somewhat richer set of errors than this prior work. The final taxonomy—incorporating feedback from a ten-language pilot study—consists of four major error categories, with several additional sub-categories. The categories are applied sequentially, as in Figure 1. We now describe these categories.

Target errors This category consists of cases where the gold data is incorrect or incomplete.[3] We discern three sub-categories of target errors. **Free variation errors** occur when more than one acceptable inflected form exists, but only one is present in the UniMorph data. **Extraction errors** indicate flaws in UniMorph's parsing of Wiktionary inflectional paradigms. **Wiktionary errors** represent errors in the Wiktionary data itself.

Silly errors This category consists of those "bizarre" errors which defy any purely linguistic characterization. In addition to the aforementioned case of **membled*, such errors have also been reported for other language generation tasks such as machine translation (Arthur et al. 2016) and text normalization (Gorman and Sproat 2016, Sproat and Jaitly 2017, Zhang et al. 2019).

Allomorphy errors This category consists of those errors which are characterized by misapplication of existing (i.e., independently attested) allomorphic patterns in the target language. Our annotation scheme recognizes four sub-categories of allomorphy error, but we set aside their their description for reasons of space.

Spelling errors This category includes inflected forms that do not follow language-specific orthographic conventions but are otherwise correct.

4 Results

We performed full error annotation on twelve of the 52 languages. Several other languages were initially targeted for annotation but produced too few errors to draw meaningful conclusions. Annotations were performed by the authors, all specialists in computational linguistics.[4] Of these, four languages—English, Finnish, Polish, and Russian—were annotated by native

[3] This label is applied regardless of whether the predicted inflected form is correct or not, and therefore is independent of system predictions. Furthermore, it is possible that both the gold data and prediction have the same incorrect inflected form, but detecting such cases is challenging.

[4] We do not claim that this level of expertise is strictly necessary; it might be the case that linguistically naïve native speakers could be trained to produce reliable annotations.

speakers; the remaining eight were annotated by second-language speakers. In addition to the annotation guidelines, annotators were encouraged to consult authoritative dictionaries and reference grammars—such as the *Iso suomen kielioppi* (Hakulinen et al. 2008) for Finnish, the *Duden* for German, the *Oxford Latin Dictionary* (Lee 1968), or the *Diccionario de la lengua española* for Spanish—and native speakers. Table 2 reports summary statistics for fully-annotated languages.

4.1 Inter-annotator agreement

Table 3 provides raw agreement and Krippendorf's α (Artstein and Poesio 2008) for those languages known to two annotators. As mentioned above, each annotator is a specialist in computational linguistics, and annotated at least one other language as well. Raw agreement is high, and while chance-corrected agreement statistics like α are notoriously difficult to interpret, $\alpha \geq 0.8$, a threshold obtained for all three double-coded languages, is generally considered to indicate substantial reliability (Krippendorff 2004:241f.).

4.2 Errors

Table 4 provides the counts of the four major categories of error for all twelve languages and for both systems. We now proceed to describe some patterns observed within these categories.

4.2.1 Target errors

Table 5 gives counts for the three sub-categories of target errors.[5]

Free variation errors Finnish has several free variation errors, many involving vowel harmony. For example, the abessive suffix has two allomorphs, namely the the back-harmonic *-tta* and the front-harmonic *-ttä*. The lemma *progestiini* 'progestin' can take the back allomorph, giving *progestiini**tta***, but, vowel harmony often fails to apply when there are many intervening neutral vowels (*i* and *e*) between the harmonic trigger and the suffix (Hakulinen et al. 2008:§17), as is the case here. Therefore, the form *progestiini**ttä***, predicted by both systems, is grammatical, though not the form given by UniMorph. Another type of free variation error affects allomorphs of the Finnish genitive plural (gen.pl.). For instance, *omeno**iden***, *omeno**itten***, *omeno**jen***, *omen**ien*** and *omena**in*** are all possible gen.pl. forms of *omena* 'apple', but only one is present in UniMorph.

Extraction errors The comparatively low accuracy on Hungarian—CLUZH-7, the best performing system on this language, achieves 89.80% per form accuracy—appears to be due to large number of extraction errors. In most cases, the error comes from pairing one paradigm cell with another cell's inflectional bundle. For instance, UniMorph incorrectly labels **lagúnák* as the accusative plural (acc.pl.) for *lagúna* 'lagoon'; it is in fact the nominative plural (nom.pl.). In Romanian, a header for the Wiktionary paradigms reading "definite articulation" is incorrectly taken as an inflected form itself! Latin also suffers from pervasive extraction errors. This language has a robust phonemic contrast between short and long monophthongs (e.g., *malus* 'unpleasant' vs. *mālus* 'apple tree'). Long monophthongs are—at least in modern editions—indicated by the macron, a horizontal line above the vowel. UniMorph extraction has somehow removed macrons from all lemmata, though they are still present in the inflected forms. Thus, systems must attempt to predict an unpredictable phonemic contrast while mapping from lemma to inflected form. As a result, the vast majority of Latin errors concern vowel length.

Wiktionary errors Errors in the Wiktionary data itself are relatively rare and largely non-systematic. For example, in Spanish, **demarce* is given as the first person singular (1sg.) present subjunctive of *demarcar* 'to demarcate' instead of the correct form *demarque*.

4.2.2 Silly errors

Silly errors were found for all languages except English; they also appear to be somewhat more common for UE-LMU-I (59) than for CLUZH-7 (37). UE-LMU-I predicts *praes**ōs*** as the acc.pl. of the Latin noun *praesul* (a title used by Roman religious leaders); there is no obvious analogue for the *ul-ōs* stem change. In German, CLUZH-7 unexpectedly truncates the gen.pl. form of the compound noun *Schädlingsbekämpfungsmittel* 'pesticide' to produce **Schädlingsbekämpfungsmit*. For the dative plural (dat.pl.) of the Russian compound noun meaning 'forced labor', UE-LMU-I inexplicably deletes the *r* of *rabóty* 'labor', giving the bizarre **prinudítel'nym **abótam***.[6] And, in Spanish,

[5] Some analyses conducted by Richard Sproat (p.c.) suggest that sub-task 2 was also highly affected by target errors.

[6] In the shared task, Russian data is given in the standard Cyrillic orthography; we have taken the liberty of romanizing

Language	Noun	Verb	Adjective	UE-LMU-I errors	CLUZH-7 errors	Overlap
Dutch	✗	✓	✓	31	32	84%
English	✗	✓	✗	28	32	24%
Finnish	✓	✓	✓	49	65	44%
German	✓	✓	✗	70	88	48%
Hungarian	✓	✓	✗	136	132	65%
Italian	✗	✓	✗	21	24	50%
Latin	✓	✓	✓	187	190	56%
Polish	✓	✓	✓	72	79	74%
Portuguese	✗	✓	✗	9	10	73%
Romanian	✓	✓	✓	109	122	59%
Russian	✓	✓	✓	84	79	60%
Spanish	✗	✓	✗	27	25	44%

Table 2: Raw error counts out of 1,000 test examples for the target languages. Checkmarks indicate whether UniMorph data was available for a given major category in that language. Error overlap is the percentage of errors made by both systems. There were 1,701 errors in total (823 from UE-LMU-I and 878 from CLUZH-7).

Language	RA	α
Dutch	0.949	0.907
English	0.861	0.855
Spanish	0.861	0.875

Table 3: Inter-annotator agreement statistics for three double-coded languages. RA: raw agreement.

UE-LMU-I gives **at**ue**ngáis* as the second person plural present subjunctive of *atener* 'to maintain'. There is no analogue for this *e-ue* stem change.

4.2.3 Allomorphy errors

With the exception of Hungarian and Latin—which suffer from systematic extraction errors—allomorphy errors are the largest category of error in all languages.

Stem-final vowels in Finnish In Finnish nouns and adjectives, stem-final vowels commonly disappear or alternate with *e* or *o* when the plural marker *i* is added to the stem (Hakulinen et al. 2008:§45). For instance, the inessive plural of *las**i*** 'glass' is *las**e**issa*. In principle, such alternations are predictable given the syllable count of the nominal stem, the stem-final consonants and the penultimate vowel, though the exact conditions are rather complex (Hakulinen et al. 2008:§46–50). For the compound noun *pohjanpystykorva* 'norrbottenspets' (a breed of dog), CLUZH-7 predicts an incorrect gen.pl. form **pohjanpystykorv**o**jen* for intended *pohjanpystykorvien*; it has transformed the stem final *a* to *o* and then selected the wrong plural marker (**-jen* instead of *-ien*) as a result.

Ablaut in Dutch and German Stem vowel alternations in the Germanic strong verbs are known as ablaut. Ablaut is robust in Dutch and German, and in both languages, is occasionally misapplied. In Dutch, for example, both systems overapply ablaut to the 1sg. preterite indicative forms of *printen* 'to print', producing **pr**o**nt* instead of *pr**i**ntte*. Similar errors are found in German. Both systems underapply ablaut to the 1sg. preterite indicative form of *saufen* 'to drink', producing **s**au**fte* instead of the expected *s**o**ff*, and UE-LMU-I overapplies ablaut to the third person preterite subjunctive of the weak verb *versenken* 'to sink', giving **vers**ä**chten* in place of the expected *vers**e**nkten*.

Umlaut in German Another stem change seen in German inflection is umlaut, which converts a *u*, *o*, or *a* in the final syllable of a stem changes to the corresponding front vowel *ü*, *ö*, or *ä*, respectively. Umlaut applies in many different morphological contexts (Hieble 1957), but most saliently in many plural nouns. One or both systems underapply umlaut in otherwise-correct plural forms of *Aasvogel* 'carrion bird', *Augenarzt* 'eye doctor', *Brunst* 'arousal', *Chalkogenidglas* 'chalcogenide glass', *Dachschaden* 'mental issues' (lit. 'roof damage'), *Energiezustand* 'energy level', *Hang* 'slope', *Stiefvater* 'stepfather', *Tibetfuchs* 'Tibetan fox', and *Vertrag* 'treaty'. But the systems also overapply umlaut in **Einwohnerzähle* (from

it here so as to make the data accessible to a wider audience.

Language	Target	Silly		Allomorphy		Spelling	
		UE-LMU-I	CLUZH-7	UE-LMU-I	CLUZH-7	UE-LMU-I	CLUZH-7
Dutch	8	1	1	19	16	5	7
English	3	0	0	18	18	7	11
Finnish	11	7	7	33	48	0	0
German	3	4	10	54	67	9	9
Hungarian	83	21	9	37	44	1	0
Italian	5	5	1	11	16	0	2
Latin	119	2	0	76	93	0	0
Polish	5	6	3	60	67	2	4
Portuguese	1	1	0	6	7	1	2
Romanian	54	3	5	61	69	1	2
Russian	7	7	0	48	45	23	28
Spanish	7	2	1	12	12	6	6
Total	306	59	37	435	502	55	71

Table 4: Error type counts by language and system; target error counts are combined across the two systems.

Language	FV	Extraction	Wiktionary
Dutch	0	3	5
English	0	2	1
Finnish	7	2	2
German	0	0	3
Hungarian	0	83	0
Italian	0	0	5
Latin	0	118	1
Polish	0	4	1
Portuguese	0	1	0
Romanian	1	51	2
Russian	1	5	1
Spanish	2	3	2
Total	11	272	23

Table 5: A breakdown of target errors by sub-category; counts are combined across the two systems. FV: free variation errors; Extraction: extraction errors; Wiktionary: Wiktionary errors.

Einwohnerzahl 'population', **Förmer* (from *Form* 'shape'), **Neuwähle* (from *Neuwahl* 're-vote'), and **Sprösse* (from *Spross* 'bud').

Consonant gradation in Finnish Many Finnish words undergo a set of unpredictable stem changes known as consonant gradation. Here, a "strong grade" of a consonant–normally a voiceless stop like *t*—alternates with the weak grade—a voiced stop like *d*—but the stop may also delete in the weak grade (Hakulinen et al. 2008:§41). Gradation leads to inflection errors because not all lexemes participate in gradation, and because the weak grade of the stem consonant is not predictable from the lemma. For instance, CLUZH-7 incorrectly applies the weak grade to the negated third person singular **ei kiemurda* (from *kiemurtaa* 'to crawl'); the proper gradation is *t-r* instead of the predicted *t-d*. CLUZH-7 also incorrectly produces the strong grade where the weak grade is required, failing to delete the *k* in the comitative **rikoslakein* (from *rikoslaki* 'criminal law').

Linking vowels in Hungarian The Hungarian noun plural suffix is *-k*, usually preceded by a *a*, *o*, *e*, or *ö* linking vowel. For example, the nom.pl. form of *vér* 'blood' is *vér**ek***. The choice of linking vowel is partly determined by vowel harmony: back vowel stems select *a* or *o* whereas front vowel stems select *e* or *ö*. However, for back vowel stems, it is largely unpredictable whether *a* or *o* is used (Siptár and Törkenczy 2000:224f., Vago 1980:110f.), and there are several cases where one or both systems predict an incorrect linking vowel. For example, UE-LMU-I predicts an incorrect elative plural **masszázsakból* for *masszázs* 'massage'; the correct form is *masszázs**ok**ból*.

Yers in Polish Another sub-category of allomorphy error in Polish concerns the yers, the "fleeting vowels" of Slavic. Oblique forms of the Polish nouns *klęsek* 'defeat' and *żag**ie**l* 'sail', for example lack a stem *e* or *ie*, respectively, in certain case forms, as seen in the gen.pl. *klęsk* and

żagli. Because fleeting vowels' position and quality are unpredictable, they cannot be analyzed as epenthetic. Instead, they are assumed to be present in the underlying form of certain roots and affixes, but somehow represented distinctly from the non-fleeting vowels (Lightner 1965, Rubach 1986). According to the analysis, a yer is deleted except when the following syllable also contains an yer, and the fleeting *e* and *ie* surface in the nom.sg. forms above because the masculine nom.sg. suffix is itself a yer (Gussman 1980:36f., Rubach 1984:41). It is impossible to predict the position or quality of a yer without referring to the rest of the inflectional paradigm,[7] and this indeterminacy contributes to several inflectional errors. For instance, CLUZH-7 predicts **żagieli* instead of the expected *żagli*, and both systems predict **klęsek* for of the expected *klęsk*. Similar errors are also found in Russian.

Spanish diphthongization Many Spanish verbs exhibit a stem change in which mid vowels *e* and *o* in the final syllable of the stem diphthongize to *ie* [je] and *ue* [we], respectively, when they bear primary stress. Whether or not a mid vowel participates in diphthongization is largely unpredictable (Brame and Bordelois 1974:132f., Harris 1969:74f.).[8] For example, *negar* 'to deny' undergoes diphthongization (e.g., 1sg. present indicative ***nie***gо), but *pegar* 'to stick' does not (1sg. present indicative *pego*). Both models underapply diphthongization in **despl**e**gue* (from *desplegar* 'to unfold') and **rec**o**la* (from *recolar* 'to strain again'). Interestingly, these are 1st conjugation (i.e., *-ar*) verbs, and children acquiring Spanish tend to underapply diphthongization in this class (Mayol 2007). But CLUZH-7 also overapplies diphthongization in **atañ**ie**res* (from *atañer* 'to concern') and **gañ**ie**seis* (from *gañir* 'to yelp'). Similar errors occur in Portuguese, which also exhibits a stress-conditioned stem vowel alternation.

Noun plural suffixes in German German has five major noun plural suffixes, and many errors involve the use of the wrong plural. The most frequent pattern is the overapplication of the *-(e)n* plural—traditionally regarded as the most productive plural suffix (Bech 1963, Wunderlich 1999)—as in *Eosin* 'eosin', *Fußballweltmeisterschaftsqualifikationsspiel* 'football world championship qualification game', *Hartung*, a poetic term for 'January', *Karbonatit* 'carbonatite', *Metallatom* 'metal atom', and *Vorjahr* 'last year'. Overapplication of *-e* is also common, as in *Abonnement* 'subscription', *Etat* 'budget', *Funke* 'spark', *Katholic* 'Catholic', *Königsgelb* 'yellow pigment', *Reaktorbau* 'reactor construction', *Prinzess* 'princess', *Toupet* 'toupee'. Interestingly, both types of error are produced by children acquiring German (e.g., Clahsen 1999, Marcus et al. 1995, Szagun 2001).

Genitive singular suffixes in Polish Polish has two gen.sg. suffixes, *-a* and *-u*. It is generally impossible to predict which gen.sg. allomorph a given stem will select, and there is no evidence that one is more productive than the other (Dąbrowska 2001, 2005, Kottum 1981, Maunsch 2003). This unpredictable allomorphy causes many gen.sg. errors to both systems, such as **ateusz**u*** for *ateusz**a*** 'atheist', **izotop**a*** for *izotop**u*** 'isotope', **krzyk**a*** for *krzyk**u*** 'scream', and **legar**u*** for *legar**a*** 'joist'.

Verbal prefixes in German Some verbal prefixes in German are known as "separable" because they separate (i.e., are postposed) from their host verb when tensed. Others, the "inseparable" prefixes, are always attached to their host verb without exception. Finally, some prefixes, such as *um-*, are separable or inseparable depending on the verb, and this leads to several errors. For example, both systems predict **__um__kehre* for the 1sg. present indicative of *umkehren* 'to turn around'; the correct form is the separable *kehre __um__*.

Animacy in Polish and Russian Case syncretisms in inanimate (i.e., non-personal) nouns are found in many Slavic languages. However, animacy is an inherent feature of nouns and cannot be predicted from the form of the lemma alone. In Russian, for example, CLUZH-7 wrongly predicts a syncretic acc.pl. for the animate *sadist* 'id.' and both systems incorrectly predict a distinct (i.e., non-syncretic) acc.sg. for the inanimate *magazin* 'shop'. Similarly in Polish, both systems predict incorrect syncretic accusatives for animates such as *śpiewak* 'singer' and *Żyd* 'Jew', and incorrect non-syncretic accusatives for inanimates such as *szampan* 'champagne'. Some Polish stem changes are also conditioned by animacy. For example, for the inanimate noun *katalizator* 'catalyst', both sys-

[7] Gouskova and Becker (2013) and Becker and Gouskova (2016) develop formal models of yer-deletion in Russian, but do not evaluate performance on actual held-out words.

[8] Albright et al. (2001) and Albright (2003) develop a computational model to predict Spanish diphthongization, but do not report its performance on actual held-out verb forms.

tems incorrectly predict a nom.pl. **katalizatorzy* instead of *katalizatory*; the mutation of *r* to *rz* before the nom.pl. *-y* is restricted to masculine animates (Feldstein 2001:27).

Aspect in Russian Russian verbal inflection is conditioned by an inherent feature known as aspect. For instance, the perfective verb *sorvat'* 'to pick' forms a synthetic future whereas the closely-related imperfective *sryvat'* forms a periphrastic (i.e., multi-word) future formed using future-tense forms of *byt'* 'to be'. Several errors involve the wrong future form for a verb's aspect. For example, for the perfective *sorvat'*, CLUZH-7 incorrectly predicts a periphrastic second person singular future **budeš' sorvat'* instead of the expected synthetic *sorvëš'*.

Vowel harmony in Finnish compounds In Finnish, the first stem in a noun compound does not participate in suffix harmony (Hakulinen et al. 2008:§14). For example, the partitive singular of the compound *lapinsirri* 'Temminck's stint' (a type of bird) is the *lapinsirriä*. Because this lemma is a compound of *Lapin* 'of Lapland' and *sirri* 'stint', and because all vowels in the second second stem of the compound are neutral, front harmony—the default—applies. However, CLUZH-7 generates **lapinsirria*, a form which would be correct were the lemma not a compound.

Internal inflection in Russian compounds Many Russian nouns in the shared task are adjective-noun or noun-noun compounds, and systems fail to appropriately inflect both components of the compound. The acc.pl. of *lëgkaja promyšlennost'* 'light industry' is *lëgkie promyšlennosti*, but UE-LMU-I predicts **lëgkix promyšlennosti*, incorrectly placing the adjective in the genitive case. Other adjective-noun compounds for which one or both of the systems fail to produce proper agreement morphology include *vizitnaja kartočka* 'business card' and *bulevo množestvo* 'boolean domain'. Both stems of most noun-noun compounds, particularly hyphenated compounds, are inflected. For example, the prepositional plural of *gosudarstvo-donor* 'donor state' is *gosudarstvax-donorax*, but both systems predict **gosudarstvo-donorax*, in which only the second stem is inflected. However, there are some cases in which one stem of a compound is not declined. For instance, in *sindrom Aspergera* 'Asperger's syndrome', only the head noun *sindrom* should be inflected because *Aspergera* is a nominal modifier and already in genitive case, but both systems incorrectly inflect the second stem producing the gen.pl. **sindromov Asperger*.

4.2.4 Spelling errors

Spelling errors are relatively rare overall. In Dutch, diaeresis is used to mark hiatus—adjacent vowels in consecutive syllables—and thus the past participle of *upgraden* 'to upgrade' should be *geüpgraded*, not the predicted **geupgraded*. Several English errors concern an orthographic doubling of certain final consonants; for example, both systems predict a past participle **disentered* instead of the expected *disenterred*. There are many German spelling errors, including several concerning the spelling of the gen.sg. suffix—written as *-es* or *-s* depending on context—or *s*, *ss*, and *ß*, all pronounced [s]. In Spanish, a *g* followed by *i* or *e* is read as [x], not as [g], so the verb *fungir* 'to service as' has a 1sg. future indicative spelled *funjo* rather than the predicted **fungo*. Several Portuguese and Spanish predictions omit the acute accent used to indicate exceptional primary stress; e.g., Portuguese **influisse* for the 1sg. imperfect subjunctive *influísse* (from *influir* 'to influence').

5 Discussion

5.1 Target errors

Target errors heavily impact performance for Hungarian, Latin, and Romanian. Overall, nearly one fourth of our sample's errors were target errors, and we suspect such errors also lurk in the training and development data. Clearly, the UniMorph data used in this task requires further vetting.

5.2 Allomorphy errors

Overall, silly errors were far less common than allomorphy errors. Many of the allomorphy errors appear to result from unpredictable linguistic behaviors rather than failures to extract reliable generalizations. In some cases, errors reflect systems' inability to predict inherent features such as animacy and aspect in Slavic. Such features are not encoded in UniMorph, although this information is often present on Wiktionary. Generally speaking, these features cannot be predicted from the orthographic form of lemmata,[9] but we

[9] Certain prefixes and stress patterns are cues to aspect in Russian verbs (Wade 2010:268), but this is not true of inherent features in general.

suspect that the relevant information could be induced using either contextual or type-level word embeddings. We leave this for future work.[10] Systems also appear to struggle with lemmata which are themselves internally complex due to word-formation processes like prefixation or compounding, including prefix verbs in German and compounds in Finnish and Russian. Lemma-internal structure, once again, is not currently encoded in UniMorph, though it could in principle be extracted from Wiktionary entries. Finally, we see that systems struggle with certain lexically-specific morphophonological patterns—Germanic ablaut and umlaut, Finnish consonant gradation, Hungarian linking vowels, Slavic yers, and Spanish diphthongization—and with lexically-conditioned affix selection in German and Polish. We have seemingly rediscovered what linguists have long known: certain allomorphic patterns cannot be predicted from the form of lemmata alone; they must be memorized. It is unreasonable to expect any neural network, no matter how powerful, to predict what is truly unpredictable.

Our analysis is limited to languages included the shared task, those for which the top systems have a non-trivial number of errors, and those for which we have sufficient linguistic expertise. As a result, our final sample of twelve languages only includes two major language families, Indo-European and Uralic, the latter represented by Finnish and Hungarian. However, this sample has some degree of grammatical diversity. Linguists traditionally distinguish between two types of morphological exponence. In **agglutination**, each morphological feature corresponds roughly to a single affix. For instance, in the Hungarian form *cinkosoknak*, the dat.pl. of *cinkos* 'accomplice', the *-ok* suffix marks plurality and the *-nak* suffix indicates the dative case. In **fusion**, on the other hand, single affixes may realize many morphological features at once. For instance, in the Russian form *čabrecov*, the gen.pl. of *čabrec* 'thyme', the *-ov* suffix is both genitive and plural (and its form also indirectly indicates that the stem is masculine). Agglutination is characteristic of the Uralic languages, whereas Indo-European languages makes heavy use of fusion. Furthermore, vowel harmony is limited to the two Uralic languages.

[10] Sub-task two of the SIGMORPHON 2019 Shared Task (McCarthy et al. 2019) involves lemmatization and morphological analysis in sentential context, but the applicability of this to the inflection task has not yet received much attention.

6 Conclusion

We propose an error taxonomy for morphological inflection generation and apply it to the predictions of the two best systems in the CoNLL–SIGMORPHON 2017 Shared Task. We estimate a lower bound for the percentage of "target" errors in the gold data. Over 80% of the remaining (non-target) errors can be understood as misapplication of language-specific morphological or spelling principles. One potential remedy is to enrich the input linguistic representations with, e.g., compound structure and inherent grammatical features; however, this is unlikely to avoid all errors; some morphological patterns cannot be generalized but only memorized.

The above analysis depends on manual annotation, but one might prefer to automate error classification. An automated system, for example, could be integrated into a rapid development process, or used as an additional objective during training and tuning, so long as it has reasonably high agreement with human experts. Ideally, such a system would scale to arbitrary languages, not just those for which linguistic expertise is readily available. A powerful ensemble model could help identify candidate target errors, and for certain high-resource languages, it might be possible to leverage finite-state morphological analyzers and lexicons to distinguish between silly, spelling, and allomorphy errors. We leave these and many other open questions for future work.

Acknowledgments

Ekaterina Levitskaya made substantial contributions to early stages of this project. Suzanne van der Feest assisted with Dutch annotations, and Alëna Aksënova assisted with Russian annotations. Richard Sproat provided spiritual guidance and contributed an impressionistic characterization of the sub-task 2 error profiles.

Miikka Silfverberg was funded in part by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 771113).

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*,

pages 2004–2015, Vancouver. Association for Computational Linguistics.

Adam Albright. 2003. A quantitative study of Spanish paradigm gaps. In *Proceedings of the 22nd West Coast Conference on Formal Linguistics*, pages 1–14, Somerville, MA. Cascadilla.

Adam Albright, Argelia Andrade, and Bruce Hayes. 2001. Segmental environments of Spanish diphthongization. *UCLA Working Papers in Linguistics*, 7:117–151.

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Austin. Association for Computational Linguistics.

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Gunnar Bech. 1963. Zur Morphologie der deutschen Substantive. *Lingua*, 12(1):177–189.

Michael Becker and Maria Gouskova. 2016. Source-oriented generalizations as grammar inference in Russian vowel deletion. *Linguistic Inquiry*, 47(3):391–425.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 31–39, Vancouver. Association for Computational Linguistics.

Jean Berko. 1958. The child's learning of English morphology. *Word*, 14:150–177.

Michael K. Brame and Ivonne Bordelois. 1974. Some controversial questions in Spanish phonology. *Linguistic Inquiry*, 5(2):282–298.

Erwin Chan. 2008. *Structures and distributions in morphology learning*. Ph.D. thesis, University of Pennsylvania.

Harald Clahsen. 1999. Lexical entries and rules of language: a multidisciplinary study of German inflection. *Behavioral and Brain Sciences*, 22(6):991–1069.

Maria Corkery, Yevgen Matusevych, and Sharon Goldwater. 2019. Are we there yet? Encoder-decoder neural networks as cognitive models of English past tense inflection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3868–3877, Florence. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27, Brussels. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. CoNLL–SIGMORPHON 2017 shared task: universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin. Association for Computational Linguistics.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig. `https://wals.info/`.

Dudenredaktion, editor. No date. *Duden in 12 Banden. 1: Die deutsche Rechtschreibung*, 27th edition. Bibliographisches Institut, Berlin.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1185–1195, Atlanta. Association for Computational Linguistics.

Ewa Dąbrowska. 2001. Learning a morphological system without a default: the Polish genitive. *Journal of Child Language*, 28(3):545–574.

Ewa Dąbrowska. 2005. Productivity and beyond: mastering the Polish genitive. *Journal of Child Language*, 32:191–205.

Ronald F. Feldstein. 2001. *A concise Polish grammar*. Slavic and East European Language Resource Center, Durham, NC.

Mark Fishel, Rico Sennrich, Maja Popović, and Ondřej Bojar. 2012. TerrorCat: a translation error categorization-based MT quality metric. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 64–70, Montréal. Association for Computational Linguistics.

Kyle Gorman and Richard Sproat. 2016. Minimally supervised number normalization. *Transactions of the Association for Computational Linguistics*, 4:507–519.

Maria Gouskova and Michael Becker. 2013. Nonce words show that Russian yer alternations are governed by the grammar. *Natural Language & Linguistic Theory*, 31(3):735–765.

Edmund Gussman. 1980. *Studies in abstract phonology*. MIT Press, Cambridge.

Auli Hakulinen, Maria Vilkuna, Riitta Korhonen, Vesa Koivisto, Tarja-Riitta Heinonen, and Irja Alho. 2008. *Iso suomen kielioppi*. Suomalaisen Kirjallisuuden Seura, Helsinki.

James Harris. 1969. *Spanish phonology*. MIT Press, Cambridge.

Jacob Hieble. 1957. What about the German umlaut? *The German Quarterly*, 30(4):272–274.

Ann Irvine, John Morgan, Marine Carpuat, Hal Daumé III, and Dragos Munteanu. 2013. Measuring machine translation errors in new domains. *Transactions of the Association for Computational Linguistics*, 1:429–440.

Katharina Kann and Hinrich Schütze. 2017. The LMU system for the CoNLL–SIGMORPHON 2017 shared task on universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 40–48, Vancouver. Association for Computational Linguistics.

Christo Kirov and Ryan Cotterell. 2018. Recurrent neural networks in linguistic theory: revisiting Pinker and Prince (1988) and the past tense debate. *Transactions of the Association for Computational Linguistics*, 6:651–665.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Arya D. McCarthy, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. UniMorph 2.0: universal morphology. In *Proceedings of the 11th Language Resources and Evaluation Conference*, pages 1868–1873, Miyazaki. European Language Resource Association.

Christo Kirov, John Sylak-Glassman, Roger Que, and David Yarowsky. 2016. Very-large scale parsing and normalization of Wiktionary morphological paradigms. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, pages 3121–3126, Portorož. European Language Resources Association.

Steiner E. Kottum. 1981. The genitive singular form of masculine nouns in Polish. *Scando-Slavica*, 27(1):179–186.

Klaus Krippendorff. 2004. *Content analysis: an introduction to its methodology*, 2nd edition. Sage, Thousand Oaks, CA.

Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. 2010. Morpho Challenge 2005–2010: evaluations and results. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, Uppsala. Association for Computational Linguistics.

G. M. Lee, editor. 1968. *Oxford Latin dictionary*. Claredon Press, Oxford.

Theodore M. Lightner. 1965. *Segmental phonology of Modern Standard Russian*. Ph.D. thesis, MIT.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 49–57, Vancouver. Association for Computational Linguistics.

Gary Marcus, Ursula Brinkmann, Harald Clahsen, Richard Wiese, and Steven Pinker. 1995. German inflection: the exception that proves the rule. *Cognitive Psychology*, 29(3):189–256.

Gary Marcus, Steven Pinker, Michael Ullman, Michelle Hollander, John Rosen, and Fei Xu. 1992. *Overregularization in language acquisition*. Monographs of the Society for Research in Child Development. University of Chicago Press, Chicago.

Hanna Maunsch. 2003. Current alternations in inflection of Polish masculine inanimate nouns in the singular: a pilot study. *Investigationes Linguisticae*, 9:4–21.

Laia Mayol. 2007. Acquisition of irregular patterns in Spanish verbal morphology. In *12th ESSLLI Student Session Proceedings*, pages 185–196, Dublin. Association for Logic, Language and Information.

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sebastian J. Mielke, Jeffrey Heinz, Ryan Cotterell, and Mans Hulden. 2019. The SIGMORPHON 2019 shared task: morphological analysis in context and cross-lingual transfer for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244, Florence. Association for Computational Linguistics.

Steven Pinker and Alan Prince. 1988. On language and connectionism: analysis of a parallel distributed processing model of language acquisition. *Cognition*, 28(1–2):73–193.

Maja Popović and Hermann Ney. 2011. Towards automatic error analysis of machine translation output. *Computational Linguistics*, 37(4):657–688.

Real Academia Española, editor. 1992. *Diccionario de la lengua española*, 21st edition. Real Academia Española, Madrid.

Alla Rozovskaya and Dan Roth. 2016. Grammatical error correction: machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2205–2215, Berlin. Association for Computational Linguistics.

Jerzy Rubach. 1984. *Cyclic and lexical phonology: the structure of Polish*. Foris, Dordrecht.

Jerzy Rubach. 1986. Abstract vowels in three-dimensional phonology: the yers. *The Linguistic Review*, 5(3):247–280.

David Rumelhart and Jay McClelland. 1986. On learning the past tenses of English verbs. In Jay McClelland, David Rumelhart, and the PDP Research Group, editors, *Parallel distributed processing: explorations into the microstructure of cognition. Vol. 2: Psychological and biological models*, pages 216–271. Bradford Books, Cambridge.

Péter Siptár and Miklós Törkenczy. 2000. *The phonology of Hungarian*. Oxford University Press, Oxford.

Richard Sproat. 1992. *Morphology and computation*. MIT Press, Cambridge.

Richard Sproat and Navdeep Jaitly. 2017. An RNN model of text normalization. In *Proceedings of Interspeech 2017*, pages 754–758, Stockholm. International Speech Communication Association.

John Sylak-Glassman. 2016. The composition and use of the universal morphological feature schema (UniMorph schema). Technical report, Department of Computer Science, Johns Hopkins University.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 674–680, Beijing. Association for Computational Linguistics.

Gisela Szagun. 2001. Learning different regularities: the acquisition of noun plurals by German-speaking children. *First Language*, 21:109–141.

Robert M. Vago. 1980. *The sound pattern of Hungarian*. Georgetown University Press, Washington, D.C.

Terence Wade. 2010. *A comprehensive Russian grammar*, 2nd edition. Wiley-Blackwell, Oxford.

Dieter Wunderlich. 1999. German noun plural reconsidered. *Behavioral & Brain Science*, 22(6):1044–1045.

Hao Zhang, Richard Sproat, Axel H. Ng., Felix Stahlberg, Xiaochang Peng, Kyle Gorman, and Brian Roark. 2019. Neural models of text normalization for speech applications. *Computational Linguistics*, 45(2):293–337.

Learning to Represent Bilingual Dictionaries

Muhao Chen[1*], **Yingtao Tian**[2*], **Haochen Chen**[2],
Kai-Wei Chang[1], **Steven Skiena**[2] & **Carlo Zaniolo**[1]
[1]University of California, Los Angeles, CA, USA
[2]The State University of New York, Stony Brook, NY, USA
muhaochen@ucla.edu; {kwchang, zaniolo}@cs.ucla.edu;
{yittian, haocchen, skiena}@cs.stonybrook.edu

Abstract

Bilingual word embeddings have been widely used to capture the correspondence of lexical semantics in different human languages. However, the cross-lingual correspondence between sentences and words is less studied, despite that this correspondence can significantly benefit many applications such as cross-lingual semantic search and textual inference. To bridge this gap, we propose a neural embedding model that leverages bilingual dictionaries[1]. The proposed model is trained to map the lexical definitions to the cross-lingual target words, for which we explore with different sentence encoding techniques. To enhance the learning process on limited resources, our model adopts several critical learning strategies, including multi-task learning on different bridges of languages, and joint learning of the dictionary model with a bilingual word embedding model. We conduct experiments on two new tasks. In the cross-lingual reverse dictionary retrieval task, we demonstrate that our model is capable of comprehending bilingual concepts based on descriptions, and the proposed learning strategies are effective. In the bilingual paraphrase identification task, we show that our model effectively associates sentences in different languages via a shared embedding space, and outperforms existing approaches in identifying bilingual paraphrases.

1 Introduction

Cross-lingual semantic representation learning has attracted significant attention recently. Various approaches have been proposed to align words of different languages in a shared embedding space (Ruder et al., 2017). By offering task-invariant semantic transfers, these approaches critically support many cross-lingual NLP tasks including neural machine translations (NMT) (Devlin et al., 2014), bilingual document classification (Zhou et al., 2016), knowledge alignment (Chen et al., 2018b) and entity linking (Upadhyay et al., 2018).

While many existing approaches have been proposed to associate lexical semantics between languages (Chandar et al., 2014; Gouws et al., 2015; Luong et al., 2015a), modeling the correspondence between lexical and sentential semantics across different languages is still an unresolved challenge. We argue that learning to represent such cross-lingual and multi-granular correspondence is well desired and natural for multiple reasons. One reason is that, learning word-to-word correspondence has a natural limitation, considering that many words do not have direct translations in another language. For example, *schadenfreude* in German, which means *a feeling of joy that comes from knowing the troubles of other people*, has no proper English counterpart word. To appropriately learn the representations of such words in bilingual embeddings, we need to capture their meanings based on the definitions.

Besides, modeling such correspondence is also highly beneficial to many application scenarios. One example is cross-lingual semantic search of concepts (Hill et al., 2016), where the lexemes or concepts are retrieved based on sentential descriptions (see Fig. 1). Others include discourse relation detection in bilingual dialogue utterances (Jiang et al., 2018), multilingual text summarization (Nenkova et al., 2012), and educational applications for foreign language learners. Finally, it is natural in foreign language learning that a human learns foreign words by looking up their meanings in the native language (Hulstijn et al., 1996). Therefore, learning such correspondence essentially mimics human learning behaviors.

[*] Both authors contributed equally to this work.

[1]We refer the term *dictionary* to its common meaning, i.e. lexical definitions of words. Note that this is different from some papers on bilingual settings that refer dictionaries to seed lexicons for one-to-one word mappings.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 152–162
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

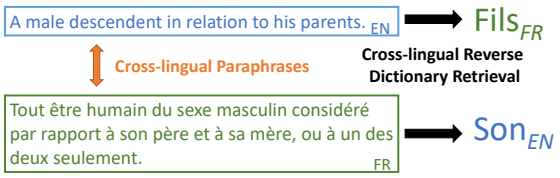


Figure 1: An example illustrating the two cross-lingual tasks. The *cross-lingual reverse dictionary retrieval* finds cross-lingual target words based on descriptions. In terms of *cross-lingual paraphrases*, the French sentence (which means *any male being considered in relation to his father and mother, or only one of them*) describes the same meaning as the English sentence, but has much more content details.

However, realizing such a representation learning model is a non-trivial task, inasmuch as it requires a comprehensive learning process to effectively compose the semantics of arbitrary-length sentences in one language, and associate that with single words in another language. Consequently, this objective also demands high-quality cross-lingual alignment that bridges between single and sequences of words. Such alignment information is generally not available in the parallel and seed-lexicon that are utilized by bilingual word embeddings (Ruder et al., 2017).

To incorporate the representations of bilingual lexical and sentential semantics, we propose an approach to capture *the mapping from the definitions to the corresponding foreign words* by leveraging *bilingual dictionaries* The proposed model `BilDRL` (**Bil**ingual **D**ictionary **R**epresentation **L**earning) first constructs a word embedding space with pre-trained bilingual word embeddings. Based on cross-lingual word definitions, a sentence encoder is trained to realize the mapping from literal descriptions to target words in the bilingual word embedding space, for which we investigate with multiple encoding techniques. To enhance cross-lingual learning on limited resources, `BilDRL` conducts multi-task learning on different directions of a language pair. Moreover, `BilDRL` enforces a joint learning strategy of bilingual word embeddings and the sentence encoder, which seeks to gradually adjust the embedding space to better suit the representation of cross-lingual word definitions.

To show the applicability of `BilDRL`, we conduct experiments on two useful cross-lingual tasks (see Fig. 1). (i) *Cross-lingual reverse dictionary retrieval* seeks to retrieve words or concepts given descriptions in another language. This task is useful to help users find foreign words based on the notions or descriptions, and is especially beneficial to users such as translators, foreigner language learners and technical writers using non-native languages. We show that `BilDRL` achieves promising results on this task, while bilingual multi-task learning and joint learning dramatically enhance the performance. (ii) *Bilingual paraphrase identification* asks whether two sentences in different languages essentially express the same meaning, which is critical to question answering or dialogue systems that apprehend multilingual utterances (Bannard and Callison-Burch, 2005). This task is challenging, as it requires a model to comprehend cross-lingual paraphrases that are inconsistent in grammar, content details and word orders. `BilDRL` maps sentences to the lexicon embedding space. This process reduces the problem to evaluate the similarity of lexicon embeddings, which can be easily solved by a simple classifier. `BilDRL` performs well with even a small amount of data, and significantly outperforms previous approaches.

2 Related Work

We discuss two lines of relevant work.

Bilingual word embeddings. Various approaches have been proposed for training bilingual word embeddings. These approaches span in two families: off-line mappings and joint training.

The off-line mapping based approach fixes the structures of pre-trained monolingual embeddings, and induces bilingual projections based on seed lexicons (Mikolov et al., 2013a). Some variants of this approach improve the quality of projections by adding constraints such as orthogonality of transforms, normalization and mean centering of embeddings (Xing et al., 2015; Artetxe et al., 2016; Vulić et al., 2016). Others adopt canonical correlation analysis to map separate monolingual embeddings to a shared embedding space (Faruqui and Dyer, 2014; Doval et al., 2018).

Unlike off-line mappings, joint training models simultaneously update word embeddings and cross-lingual alignment. In doing so, such approaches generally capture more precise cross-lingual semantic transfer (Ruder et al., 2017; Upadhyay et al., 2018). While a few such models still maintain separated embedding spaces for each language (Artetxe et al., 2017), more of them maintain a unified space for both languages. The

cross-lingual semantic transfer by these models is captured from parallel corpora with sentential or document-level alignment, using techniques such as bilingual bag-of-words distances (BilBOWA) (Gouws et al., 2015), Skip-Gram (Coulmance et al., 2015) and sparse tensor factorization (Vyas and Carpuat, 2016).

Neural sentence modeling. Neural sentence models seek to capture phrasal or sentential semantics from word sequences. They often adopt encoding techniques such as recurrent neural encoders (RNN) (Kiros et al., 2015), convolutional encoders (CNN) (Chen et al., 2018a), and attentive encoders (Rocktäschel et al., 2016) to represent the composed semantics of a sentence as an embedding vector. Recent works have focused on apprehending pairwise correspondence of sentential semantics by adopting multiple neural sentence models in one learning architecture, including Siamese models for detecting discourse relations of sentences (Sha et al., 2016), and sequence-to-sequence models for tasks like style transfer (Shen et al., 2017), text summarization (Chopra et al., 2016) and translation (Wu et al., 2016).

On the other hand, fewer efforts have been put to characterizing the associations between sentential and lexical semantics. Hill et al. (2016) and Ji et al. (2017) learn off-line mappings between monolingual descriptions and lexemes to capture such associations. Eisner et al. (2016) adopt a similar approach to capture emojis based on descriptions. At the best of our knowledge, there has been no previous approach to learn to discover the correspondence of sentential and lexical semantics in a multilingual scenario. This is exactly the focus of our work, in which the proposed strategies of multi-task learning and joint learning are critical to the corresponding learning process under limited resources. Utilizing such correspondence, our approach also sheds light on addressing discourse relation detection in a multilingual scenario.

3 Modeling Bilingual Dictionaries

We hereby begin our modeling with the formalization of bilingual dictionaries. We use $\mathcal{L}$ to denote the set of languages. For a language $l \in \mathcal{L}$, V_l denotes its vocabulary, where for each word $w \in V_l$, bold-faced $\mathbf{w} \in \mathbb{R}^k$ denotes its embedding vector. A l_i-l_j bilingual dictionary $D(l_i, l_j)$ (or simply D_{ij}) contains dictionary entries $(w^i, S_w^j) \in D_{ij}$, in which $w^i \in V_{l_i}$, and $S_w^j = w_1^j \dots w_n^j$ ($w_\cdot^j \in V_{l_j}$) is a cross-lingual definition that describes the word w^i with a sequence of words in language l_j. For example, a French-English dictionary $D(\mathrm{Fr}, \mathrm{En})$ could include a French word *appétite* accompanied by its English definition *desire for, or relish of food or drink*. Note that, for a word w^i, multiple definitions in l_j may coexist.

`BilDRL` is constructed and improved through three stages, as depicted in Fig. 2. A sentence encoder is first used to learn from a bilingual dictionary the association between words and definitions. Then in a pre-trained bilingual word embedding space, multi-task learning is conducted on both directions of a language pair. Lastly, joint learning with word embeddings is enforced to simultaneously adjust the embedding space during the training of the dictionary model, which further enhances the cross-lingual learning process.

It is noteworthy that, NMT (Wu et al., 2016) is considered as an ostensibly relevant method to ours. NMT does not apply to our problem setting bacause it has major differences from our work in those perspectives: (i) In terms of data modalities, NMT has to bridge between corpora of the same granularity, i.e. either between sentences or between lexicons. This is unlike `BilDRL` that captures multi-granular correspondence of semantics across different modalities, i.e. sentences and words; (ii) As for learning strategies, NMT relies on an encoder-decoder architecture using end-to-end training (Luong et al., 2015b), while `BilDRL` employs joint learning of a dictionary-based sentence encoder and a bilingual embedding space.

3.1 Encoders for Lexical Definitions

`BilDRL` models a dictionary using a neural sentence encoder $E(S)$, which composes the meaning of the sentence into a latent vector representation. We hereby introduce this model component, which is designed to be a GRU encoder with self-attention. Besides that, we also investigate other widely-used neural sequence encoders.

3.1.1 Attentive GRU Encoder

The GRU encoder is a computationally efficient alternative of the LSTM (Cho et al., 2014). Each unit consists of a reset gate $\mathbf{r}_t$ and an update gate $\mathbf{z}_t$ to track the state of the sequence. Given the vector representation $\mathbf{w}_t$ of an incoming item w_t, GRU updates the hidden state $\mathbf{h}_t^{(1)}$ as a linear combination of the previous state $\mathbf{h}_{t-1}^{(1)}$ and the candidate

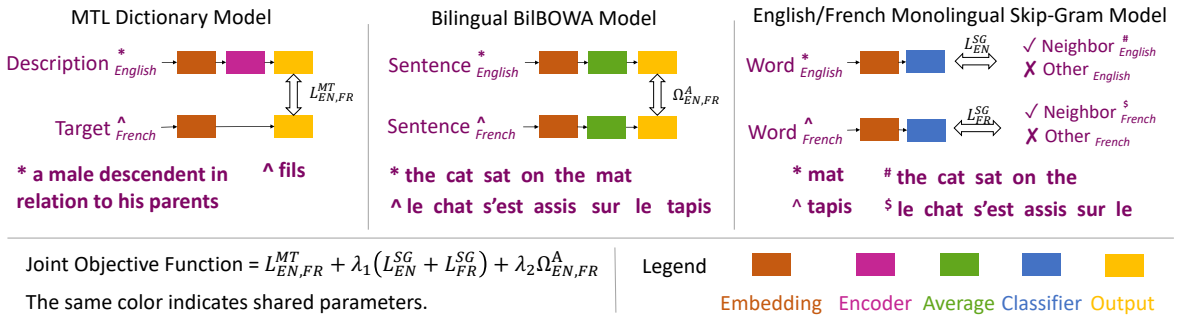


Figure 2: Joint learning architecture of BilDRL.

state $\tilde{\mathbf{h}}_t^{(1)}$ of the new item w_t as below.

$$\mathbf{h}_t^{(1)} = \mathbf{z}_t \odot \tilde{\mathbf{h}}_t^{(1)} + (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1}^{(1)}.$$

The update gate $\mathbf{z}_t$ balances between the information of the previous sequence and the new item, where $\mathbf{M}_z$ and $\mathbf{N}_z$ are two weight matrices, $\mathbf{b}_z$ is a bias vector, and σ is the sigmoid function.

$$\mathbf{z}_t = \sigma\left(\mathbf{M}_z\mathbf{x}_t + \mathbf{N}_z\mathbf{h}_{t-1}^{(1)} + \mathbf{b}_z\right).$$

The candidate state $\tilde{\mathbf{h}}_t^{(1)}$ is calculated similarly to those in a traditional recurrent unit as below. The reset gate $\mathbf{r}_t$ thereof controls how much information of the past sequence should contribute to the candidate state.

$$\tilde{\mathbf{h}}_t^{(1)} = \tanh\left(\mathbf{M}_s\mathbf{w}_t + \mathbf{r}_t \odot (\mathbf{N}_s\mathbf{h}_{t-1}^{(1)}) + \mathbf{b}_s\right)$$
$$\mathbf{r}_t = \sigma\left(\mathbf{M}_r\mathbf{w}_t + \mathbf{N}_r\mathbf{h}_{t-1}^{(1)} + \mathbf{b}_r\right).$$

While a GRU encoder can stack multiple of the above GRU layers, without an attention mechanism, the last state $\mathbf{h}_S^{(1)}$ of the last layer represents the overall meaning of the encoded sentence S.

The self-attention mechanism (Conneau et al., 2017) seeks to highlight the important units in an input sentence when capturing its overall meaning, which is calculated as below:

$$\mathbf{u}_t = \tanh\left(\mathbf{M}_a\mathbf{h}_t^{(1)} + \mathbf{b}_a\right)$$
$$a_t = \frac{\exp\left(\mathbf{u}_t^\top \mathbf{u}_S\right)}{\sum_{w_m \in S} \exp\left(\mathbf{u}_m^\top \mathbf{u}_S\right)}$$
$$\mathbf{h}_t^{(2)} = |S| a_t \mathbf{u}_t.$$

$\mathbf{u}_t$ is the intermediary representation of GRU output $\mathbf{h}_t^{(1)}$, and $\mathbf{u}_S = \tanh(\mathbf{M}_a\mathbf{h}_S^{(1)} + \mathbf{b}_a)$ is that of the last GRU output $\mathbf{h}_S^{(1)}$. $\mathbf{u}_S$ can be seen as a high-level representation of the input sequence. By measuring the similarity of each $\mathbf{u}_t$ with $\mathbf{u}_S$, the normalized attention weight a_t, which highlights an input that contributes significantly to the overall meaning, is produced through a softmax. Note that a scalar $|S|$ is multiplied along with a_t to $\mathbf{u}_t$, so as to keep the weighted representation $\mathbf{h}_t^{(2)}$ from losing the scale of $\mathbf{h}_t^{(1)}$. The sentence encoding is calculated as the average of the last attention layer $E^{(1)}(S) = \frac{1}{|S|}\sum_{t=1}^{|S|} a_t\mathbf{h}_t^{(2)}$.

3.1.2 Other Encoders

We also experiment with other widely used neural sentence modeling techniques[2], which are however outperformed by the attentive GRU in our tasks. These techniques include the vanilla GRU, CNN (Kalchbrenner et al., 2014), and linear bag-of-words (BOW) (Hill et al., 2016). We briefly introduce the later two techniques in the following.

A convolutional encoder applies a kernel $\mathbf{M}_c \in \mathbb{R}^{h\times k}$ to produce the latent representation $\mathbf{h}_t^{(3)} = \tanh(\mathbf{M}_c\mathbf{w}_{t:t+h-1} + \mathbf{b}_c)$ from each h-gram of the input vector sequence $\mathbf{w}_{t:t+h-1}$, for which h is the kernel size and $\mathbf{b}_c$ is a bias vector. A sequence of latent vectors $\mathbf{H}^{(3)} = [\mathbf{h}_1^{(3)}, \mathbf{h}_2^{(3)}, ..., \mathbf{h}_{|S|-h+1}^{(3)}]$ is produced from the input, where each latent vector leverages the significant local semantic features from each h-gram. Following convention (Liu et al., 2017), we apply dynamic max-pooling to extract robust features from the convolution outputs, and use the mean-pooling results of the last layer to represent the sentential semantics.

The Linear bag-of-words (BOW) encoder (Ji et al., 2017; Hill et al., 2016) is realized by the

[2]Note that recent advances in monolingual contextualized embeddings like multilingual ELMo (Peters et al., 2018; Che et al., 2018) and M-BERT (Pires et al., 2019; Devlin et al., 2018) can also be supported to represent sentences for our setting. We leave them as future work, as they require non-trivial adaption to both multilingual settings and joint training, and extensive pre-training on external corpora.

sum of projected word embeddings of the input sentence, i.e. $E^{(2)}(S) = \sum_{t=1}^{|S|} \mathbf{M}_b \mathbf{w}_t$.

3.2 Basic Learning Objective

The objective of learning the dictionary model is to map the encodings of cross-lingual word definitions to the target word embeddings. This is realized by minimizing the following L_2 loss,

$$L_{ij}^{\mathrm{ST}} = \frac{1}{|D_{ij}|} \sum_{(w^i, S_w^j) \in D_{ij}} \left\| E_{ij}(S_w^j) - \mathbf{w}^i \right\|_2^2$$

in which E_{ij} is the dictionary model that maps from descriptions in l_j to words in l_i.

The above defines the basic model variants of `BilDRL` that learns on a single dictionary. For word representations in the learning process, `BilDRL` initializes the embedding space using pre-trained word embeddings. Note that, without adopting the joint learning strategy in Section 3.4, the learning process does not update word embeddings that are used to represent the definitions and target words. While other forms of loss such as cosine proximity (Hill et al., 2016) and hinge loss (Ji et al., 2017) may also be used in the learning process, we find that L_2 loss consistently leads to better performance in our experiments.

3.3 Bilingual Multi-task Learning

In cases where entries in a bilingual dictionary are not amply provided, learning the above bilingual dictionary on one ordered language pair may fall short in insufficiency of alignment information. One practical solution is to conduct a bilingual multi-task learning process. In detail, given a language pair (l_i, l_j), we learn the dictionary model E_{ij} on both dictionaries D_{ij} and D_{ji} with shared parameters. Correspondingly, we rewrite the previous learning objective function as below, in which $D = D_{ij} \cup D_{ji}$.

$$L_{ij}^{\mathrm{MT}} = \frac{1}{|D|} \sum_{(w, S_w) \in D} \left\| E_{ij}(S_w) - \mathbf{w} \right\|_2^2 .$$

This strategy non-trivially requests the same dictionary model to represent semantic transfer in two directions of the language pair. To fulfill such a request, we initialize the embedding space using the BilBOWA embeddings (Gouws et al., 2015), which provide a unified embedding space that resolves both monolingual and cross-lingual semantic relatedness of words. In practice, we find this simple multi-task strategy to bring significant improvement to our cross-lingual tasks. Note that, besides BilBOWA, other joint-training bilingual embeddings in a unified space (Doval et al., 2018) can also support this strategy, for which we leave the comparison to future work.

3.4 Joint Learning Objective

While above learning strategies are based on a fixed embedding space, we lastly propose a joint learning strategy. During the training process, this strategy simultaneously updates the embedding space based on both the dictionary model and the bilingual word embedding model. The learning is through asynchronous minimization of the following joint objective function,

$$J = L_{ij}^{\mathrm{MT}} + \lambda_1 (L_i^{\mathrm{SG}} + L_j^{\mathrm{SG}}) + \lambda_2 \Omega_{ij}^{\mathrm{A}},$$

where λ_1 and λ_2 are two positive hyperparameters. L_i^{SG} and L_j^{SG} are the original Skip-Gram losses (Mikolov et al., 2013b) to separately obtain word embeddings on monolingual corpora of l_i and l_j. Ω_{ij}^{A}, termed as below, is the alignment loss to minimize bag-of-words distances for aligned sentence pairs (S^i, S^j) in parallel corpora C_{ij}.

$$\Omega_{ij}^{\mathrm{A}} = \frac{1}{|C_{ij}|} \sum_{(S^i, S^j) \in C_{ij}} d_{\mathrm{S}}(S^i, S^j)$$

$$d_{\mathrm{S}}(S^i, S^j) = \left\| \frac{1}{|S^i|} \sum_{w_m^i \in S^i} \mathbf{w}_m^i - \frac{1}{|S^j|} \sum_{w_n^j \in S^j} \mathbf{w}_n^j \right\|_2^2$$

The joint learning process adapts the embedding space to better suit the dictionary model, which is shown to further enhance the cross-lingual learning of `BilDRL`.

3.5 Training

To initialize the embedding space, we pre-trained BilBOWA on the parallel corpora Europarl v7 (Koehn, 2005) and monolingual corpora of tokenized Wikipedia dump (Al-Rfou et al., 2013). For models without joint learning, we use AMSGrad (Reddi et al., 2018) to optimize the parameters. Each model without bilingual multi-task learning thereof, is trained on batched samples from each individual dictionary. Multi-task learning models are trained on batched samples from two dictionaries. Within each batch, entries of different directions of languages can be mixed together. For joint learning, we conduct an efficient

multi-threaded asynchronous training (Mnih et al., 2016) of AMSGrad. In detail, after initializing the embedding space based on pre-trained BilBOWA, parameter updating based on the four components of J occurs across four worker threads. Two monolingual threads select batches of monolingual contexts from the Wikipedia dump of two languages for Skip-Gram, one alignment thread randomly samples parallel sentences from Europarl, and one dictionary thread extracts samples of entries for a bilingual multi-task dictionary model. Each thread makes a batched update to model parameters asynchronously for each term of J. The asynchronous training of all threads keeps going until the dictionary thread finishes its epochs.

4 Experiments

We present experiments on two multilingual tasks: the cross-lingual reverse dictionary retrieval task and the bilingual paraphrase identification task.

4.1 Datasets

The experiment of cross-lingual reverse dictionary retrieval is conducted on a trilingual dataset *Wikt3l*. This dataset is extracted from Wiktionary[3], which is one of the largest freely available multilingual dictionary resources on the Web. Wikt3l contains dictionary entries of language pairs (English, French) and (English, Spanish), which form En-Fr, Fr-En, En-Es and Es-En dictionaries on four bridges of languages in total. Two types of cross-lingual definitions are extracted from Wiktionary: (i) cross-lingual definitions provided under the *Translations* sections of Wiktionary pages; (ii) monolingual definitions for words that are linked to a cross-lingual counterpart with a inter-language link[4] of Wiktionary. We exclude all the definitions of stop words in constructing the dataset, and list the statistics in Table 1.

Since existing datasets for paraphrase identification are merely monolingual, we contribute with another dataset *WBP3l* for cross-lingual sentential paraphrase identification. This dataset contains 6,000 pairs of bilingual sentence pairs respectively for En-Fr and En-Es settings. Within each bilingual setting, positive cases are formed as pairs of descriptions aligned by inter-language links, which exclude the word descriptions in Wikt3l for training `BilDRL`. To generate negative examples, given a source word, we first find its 15 nearest neighbors in the embedding space. Within the nearest neighbors, we use ConceptNet (Speer et al., 2017) to filter out the synonyms of the source word, so as to prevent from generating false negative cases. Then we randomly pick one word from the filtered neighbors and pair its cross-lingual definition with the English definition of the source word to create a negative case. This process ensures that each negative case is endowed with limited dissimilarity of sentence meanings, which makes the decision more challenging. For each language setting, we randomly select 70% for training, 5% for validation, and the rest 25% for testing. Note that each language setting of this dataset thereof, matches with the quantity and partitioning of sentence pairs in the widely-used Microsoft Research Paraphrase Corpus benchmark for monolingual paraphrase identification (Yin et al., 2016; Das and Smith, 2009). Several examples from the dataset are shown in Table 2. The datasets and the processing scripts are available at `https://github.com/muhaochen/bilingual_dictionaries`.

Dictionary	En-Fr	Fr-En	En-Es	Es-En
#Target words	15,666	16,857	8,004	16,986
#Definitions	50,412	58,808	20,930	56,610

Table 1: Statistics of the bilingual dictionary dataset Wikt3l.

Positive Examples
En:*Being remote in space.* **Fr**:*Se trouvant à une grande distance.*
En:*The interdisciplinary science that applies theories and methods of the physical sciences to questions of biology.* **Es**:*Ciencia que emplea y desarrolla las teorias y métodos de la física en la investigación de los sistemas biolgicos.*
Negative Examples
En:*A person who secedes or supports secession from a political union.* **Fr**:*Contrôle politique exercé par une grande puissance sur une contre inféodée.*
En:*The fear of closed, tight places.* **Es**:*Pérdida o disminución considerables de la memoria.*

Table 2: Examples of bilingual paraphrases from WBP3l.

4.2 Cross-lingual Reverse Dictionary Retrieval

The objective of this task is to enable cross-lingual semantic retrieval of words based on descriptions. Besides comparing variants of `BilDRL` that adopt different sentence encoders and learning strate-

[3] `https://www.wiktionary.org/`

[4] An inter-language link matches the entries of counterpart words between language versions of Wiktionary.

Languages	En-Fr			Fr-En			En-Es			Es-En		
Metric	$P@1$	$P@10$	MRR	$P@1$	$P@10$	MRR	$P@1$	$P@10$	MRR	$P@1$	$P@10$	MRR
BOW	0.8	3.4	0.011	0.4	2.2	0.006	0.4	2.4	0.007	0.4	2.6	0.007
CNN	6.0	12.4	0.070	6.4	14.8	0.072	3.8	7.2	0.045	7.0	16.8	0.088
GRU	35.6	46.0	0.380	38.8	49.8	0.410	47.8	59.0	0.496	57.6	67.2	0.604
ATT	38.8	47.4	0.411	39.8	50.2	0.425	51.6	59.2	0.534	60.4	68.4	0.629
GRU-mono	21.8	33.2	0.242	27.8	37.0	0.297	34.4	41.2	0.358	36.8	47.2	0.392
ATT-mono	22.8	33.6	0.249	27.4	39.0	0.298	34.6	42.2	0.358	39.4	48.6	0.414
GRU-MTL	43.4	49.2	0.452	44.4	52.8	0.467	50.4	60.0	0.530	63.6	71.8	0.659
ATT-MTL	46.8	56.6	0.487	47.6	56.6	0.497	55.8	62.2	0.575	66.4	75.0	0.687
ATT-joint	**63.6**	**69.4**	**0.654**	**68.2**	**75.4**	**0.706**	**69.0**	**72.8**	**0.704**	**78.6**	**83.4**	**0.803**

Table 3: Cross-lingual reverse dictionary retrieval results by BilDRL variants. We report $P@1$, $P@10$, and MRR on four groups of models: (i) basic dictionary models that adopt four different encoding techniques (BOW, CNN, GRU and ATT); (ii) models with the two best encoding techniques that enforce the monolingual retrieval approach by Hill et al. (2016) (GRU-mono and ATT-mono); (iii) models adopting bilingual multi-task learning (GRU-MTL and ATT-MTL); (iv) joint learning that employs the best dictionary model of ATT-MTL (ATT-joint).

gies, we also compare with the monolingual retrieval approach proposed by Hill et al. (2016). Instead of directly associating cross-lingual word definitions, this approach learns definition-to-word mappings in a monolingual scenario. When it applies to the multilingual setting, given a lexical definition, it first retrieves the corresponding word in the source language. Then, it looks for semantically related words in the target language using bilingual word embeddings. As discussed in Section 3, NMT does not apply to this task due that it cannot capture the multi-granular correspondence between a sentence and a word.

Evaluation Protocol. Before training the models, we randomly select 500 word definitions from each dictionary respectively as test cases, and exclude these definitions from the training data. Each of the basic BilDRL variants are trained on one bilingual dictionary. The monolingual retrieval models are trained to fit the target words in the original languages of the word definitions, which are also provided in Wiktionary. BilDRL variants with multi-task or joint learning use both dictionaries of the same language pair. In the test phase, for each test case $(w^i, S_w^j) \in D_{ij}$, the prediction performs a kNN search from the definition encoding $E_{ij}(S_w^j)$, and record the rank of w^i within the vocabulary of l_i. We limit the vocabularies to all words that appear in the Wikt3l dataset, which involve around 45k English words, 44k French words and 36k Spanish words. To prevent the surface information of the target word from appearing in the definition, we have also masked out any translation of the target word occurring in the definition using a wildcard token <concept>. We aggregate three metrics on test cases: the accuracy $P@1$ (%), the proportion of ranks no larger than 10 $P@10$ (%), and mean reciprocal rank MRR.

We pre-train BilBOWA based on the original configuration by Gouws et al. (2015) and obtain 50-dimensional initialization of bilingual word embedding spaces respectively for the English-French and English-Spanish settings. For CNN, GRU, and attentive GRU (ATT) encoders, we stack five of each corresponding encoding layers with hidden-sizes of 200, and two affine layers are applied to the final output for dimension reduction. This encoder architecture consistently represents the best performance through our tuning. Through comprehensive hyperparameter tuning, we fix the learning rate α to 0.0005, the exponential decay rates of AMSGrad β_1 and β_2 to 0.9 and 0.999, coefficients λ_1 and λ_2 to both 0.1, and batch size to 64. Kernel-size and pooling-size are both set to 2 for CNN. Word definitions are zero-padded (short ones) or truncated (long ones) to the sequence length of 15, since most definitions (over 92%) are within 15 words in the dataset. Training is limited to 1,000 epochs for all models as well as the dictionary thread of asynchronous joint learning, in which all models are able to converge.

Results. Results are reported in Table 3 in four groups. The first group compares four different encoding techniques for the basic dictionary models. GRU thereof consistently outperforms CNN and BOW, since the latter two fail to capture the important sequential information for descriptions. ATT that weighs among the hidden states has notable improvements over GRU. While we equip the two better encoding techniques with the monolingual retrieval approach (GRU-mono and ATT-mono), we find that the way of learning the dictionary models towards monolingual targets and

retrieving cross-lingual related words incurs more impreciseness to the task. For models of the third group that conduct multi-task learning in two directions of a language pair, the results show significant enhancement of performance in both directions. For the final group of results, we incorporate the best variant of multi-task models into the joint learning architecture, which leads to compelling improvement of the task on all settings. This demonstrates that properly adapting the word embeddings in joint with the bilingual dictionary model efficaciously constructs the embedding space that suits better the representation of both bilingual lexical and sentential semantics.

In general, this experiment has identified the proper encoding techniques of the dictionary model. The proposed strategies of multi-task and joint learning effectively contribute to the precise characterization of the cross-lingual correspondence of lexical and sentential semantics, which have led to very promising capability of cross-lingual reverse dictionary retrieval.

4.3 Bilingual Paraphrase Identification

The bilingual paraphrase identification problem[5] is a binary classification task with the goal to decide whether two sentences in different languages express the same meanings. BilDRL provides an effective solution by transferring sentential meanings to word-level representations and learning a simple classifier. We evaluate three variants of BilDRL on this task using WBP3l: the multi-task BilDRL with GRU encoders (BilDRL-GRU-MTL), the multi-task BilDRL with attentive GRU encoders (BilDRL-ATT-MTL), and the joint learning BilDRL with with attentive GRU encoders (BilDRL-ATT-joint). We compare against several baselines of neural sentence pair models that are proposed for monolingual paraphrase identification. These models include Siamese structures of CNN (BiCNN) (Yin and Schütze, 2015), RNN (BiLSTM) (Mueller and Thyagarajan, 2016), attentive CNN (ABCNN) (Yin et al., 2016), attentive GRU (BiATT) (Rocktäschel et al., 2016), and BOW (BiBOW). To support the reasoning of cross-lingual semantics, we provide the baselines with the same BilBOWA embeddings.

[5]Paraphrases have similar meanings, but can largely differ in content details and word orders. Hence, they are essentially different from translations. We have found that even the well-recognized Google NMT frequently caused distortions to short sentence meanings, and led to results that were close to random guess by the baseline classifiers after translation.

Languages	En&Fr		En&Es	
Metrics	*Acc.*	*F1*	*Acc.*	*F1*
BiBOW	54.93	0.622	56.27	0.623
BiCNN	54.33	0.625	53.80	0.611
ABCNN	56.73	0.644	58.83	0.655
BiLSTM	59.60	0.662	57.60	0.637
BiATT	61.47	0.699	61.27	0.689
BilDRL-GRU-MTL	64.80	0.732	63.33	0.722
BilDRL-ATT-MTL	65.27	0.735	66.07	0.735
BilDRL-ATT-joint	**68.53**	**0.785**	**67.13**	**0.759**

Table 4: Accuracy and F1-scores of bilingual paraphrase identification. For BilDRL, the results by three model variants are reported: BilDRL-GRU-MTL and BilDRL-ATT-MTL are models with bilingual multi-task learning, and BilDRL-ATT-joint is the best ATT-based dictionary model variant deployed with both multi-task and joint learning.

Evaluation protocol. BilDRL transfers each sentence into a vector in the word embedding space. Then, for each sentence pair in the train set, a Multi-layer Perceptron (MLP) with a binary softmax loss is trained on the subtraction of two vectors as a downstream classifier. Baseline models are trained end-to-end, each of which directly uses a parallel pair of encoders with shared parameters and an MLP that is stacked to the subtraction of two sentence vectors. Note that some works use concatenation (Yin and Schütze, 2015) or Manhattan distances (Mueller and Thyagarajan, 2016) of sentence vectors instead of their subtraction (Jiang et al., 2018), which we find to be less effective on small amount of data.

We apply the configurations of the sentence encoders from the last experiment to corresponding baselines, so as to show the performance under controlled variables. Training of a classifier is terminated by early-stopping based on the validation set. Following convention (Hu et al., 2014; Yin et al., 2016), we report the accuracy and F1 scores.

Results. This task is challenging due to the heterogeneity of cross-lingual paraphrases and limitedness of learning resources. The results in Table 4 show that all the baselines, where BiATT consistently outperforms the others, merely reaches slightly over 60% of accuracy on both En-Fr and En-Es settings. We believe that it comes down to the fact that sentences of different languages are often drastically heterogenous in both lexical semantics and the sentence grammar that governs the composition of words. Hence, it is not surprising that previous neural sentence pair models, which capture the semantic relation of bilingual sentences directly from all participating words, fall short at the multilingual task. BilDRL, how-

ever, effectively leverages the correspondence of lexical and sentential semantics to simplify the task to an easier entailment task in the lexicon space, for which the multi-task learning BilDRL-ATT-MTL outperforms the best baseline respectively by 3.80% and 4.80% of accuracy in both language settings, while BilDRL-ATT-joint, employing the joint learning, further improves the task by another satisfying 3.26% and 1.06% of accuracy. Both also show notable increment in F1.

5 Conclusion and Future Work

In this paper, we propose a neural embedding model BilDRL that captures the correspondence of cross-lingual lexical and sentential semantics. We experiment with multiple forms of neural models and identify the best technique. The two learning strategies, bilingual multi-task learning and joint learning, are effective at enhancing the cross-lingual learning with limited resources, and also achieve promising performance on cross-lingual reverse dictionary retrieval and bilingual paraphrase identification tasks by associating lexical and sentential semantics. An important direction of future work is to explore whether the word-sentence alignment can improve bilingual word embeddings. Applying BilDRL to bilingual question answering and semantic search systems is another important direction.

6 Acknowledgement

We thank the anonymous reviewers for their insightful comments. This work was supported in part by National Science Foundation Grant IIS-1760523.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Plyglot: Distributed word representations for multilingual nlp. In *The SIGNLL Conference on Computational Natural Language Learning*.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 451–462.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics.

Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *CoNLL Shared Task*, pages 55–64. ACL.

Muhao Chen, Chang Ping Meng, Gang Huang, and Carlo Zaniolo. 2018a. Neural article pair modeling for wikipedia sub-article machine. In *Proceedings of European Conference of Machine Learning*.

Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Carlo Zaniolo. 2018b. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, fast cross-lingual word-embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1109–1113.

Dipanjan Das and Noah A Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing*, pages 468–476. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1370–1380.

Yerai Doval, Jose Camacho-Collados, Luis Espinosa Anke, and Steven Schockaert. 2018. Improving cross-lingual word embeddings by meeting in the middle. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 294–304.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 748–756.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.

Jan H Hulstijn, Merel Hollander, and Tine Greidanus. 1996. Incidental vocabulary learning by advanced foreign language students: The influence of marginal glosses, dictionary use, and reoccurrence of unknown words. *The modern language journal*, 80(3):327–339.

Guoliang Ji, Kang Liu, Shizhu He, Jun Zhao, et al. 2017. Distant supervision for relation extraction with sentence-level attention and entity descriptions. In *Proceedings of the AAAI International Conference on Artificial Intelligence*, pages 3060–3066.

Jyun-Yu Jiang, Francine Chen, Yan-Ying Chen, and Wei Wang. 2018. Learning to disentangle interleaved conversational threads with a siamese hierarchical network and similarity ranking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1812–1822.

Nal Kalchbrenner, Edward Grefenstette, Phil Blunsom, Dimitri Kartsaklis, Nal Kalchbrenner, Mehrnoosh Sadrzadeh, Nal Kalchbrenner, Phil Blunsom, Nal Kalchbrenner, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 212–217. Association for Computational Linguistics.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015a. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, et al. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning.

In *International Conference on Machine Learning*, pages 1928–1937.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2786–2792.

Ani Nenkova, Kathleen McKeown, et al. 2012. A survey of text summarization techniques. In *Mining text data*, pages 43–76. Springer.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *ACL*.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *International Conference on Learning Representations*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *International Conference on Learning Representations*.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. *Journal of Artificial Intelligence Research*.

Lei Sha, Baobao Chang, et al. 2016. Reading and thinking: Re-read lstm unit for textual entailment recognition. In *Proceedings of the International Conference on Computational Linguistics*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Robert Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Shyam Upadhyay, Nitish Gupta, and Dan Roth. 2018. Joint multilingual supervision for cross-lingual entity linking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2486–2495.

Ivan Vulić, Anna Korhonen, et al. 2016. On the role of seed lexicons in learning bilingual word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 247–257.

Yogarshi Vyas and Marine Carpuat. 2016. Sparse bilingual word representations for cross-lingual lexical entailment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1187–1197.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1006–1011.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Wenpeng Yin, Hinrich Schütze, et al. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4(1).

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Cross-lingual sentiment classification with bilingual document representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1403–1412.

Improving Natural Language Understanding by Reverse Mapping Bytepair Encoding

Chaodong Tong[1,2] **Huailiang Peng**[1,2](✉) **Qiong Dai**[1,2] **Lei Jiang**[1,2] **Jianghua Huang**[3]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Meituan-Dianping Group, Beijing, China
{tongchaodong, penghuailiang, daiqiong, jianglei}@iie.ac.cn
{huangjianghua}@meituan.com

Abstract

Recently, language models (LMs) or language representation models are widely used in natural language understanding (NLU) tasks. However, these LMs are usually trained on large unlabeled text corpora, while the fine-tuning process simply takes words or word-pieces as model input. Because of the differences between language model and NLU task objectives, the problem of lack of concern on some key words exists. Thus in this paper, we propose a method called reverse mapping bytepair encoding, which maps named-entity information and other word-level linguistic features back to subwords during the encoding procedure of bytepair encoding (BPE). We employ this method to the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018) by adding a weighted linear layer after the embedding layer. We also propose a new model architecture named as the multi-channel separate transformer to evaluate the effectiveness of the newly introduced information by employing a training process without parameter-sharing. Experiments on Story Cloze, RTE, SciTail and SST-2 datasets demonstrate the effectiveness of our approach. Compared with the original results in GPT, our approach gains 1.58% absolute increase on Stories Cloze, 6.4% on RTE, 0.69% on SciTail and 0.8% on SST-2.

1 Introduction

Recently, language models are widely used as the feature extractor for many NLU tasks. Statistical language models learn the joint probability function of sequences of words in a language (Bengio et al., 2003). Trained on large corpora and different domains give LMs generalization abilities, and enable them to capture latent or deep senmantics. Normally, statistical language models take the following objective to maximize:

$$L_1(u) = \sum_i \log P(u_i | u_{i-k}, ..., u_{i-1}; \Theta) \quad (1)$$

where k is the size of the context window, and the conditional probability P is modeled using a neural network with parameters Θ.

However, this formal simplicity determines that they can not deal well with the ambiguity of words nor low-frequency words. In fact, low-frequency words may appear once or little times in the whole corpus, thus the embeddings of them may overfit or underfit the corpus. In practice, we usually truncate them from the vocabulary and replace them with an "UNK" label. In this situation, we partially lose their meanings. For example, in "15 million tonnes of rubbish are produced daily in Cairo.", the number "15" can hardly be trained to gain its proper representation even if it is replaced by a specific label representing numbers. Besides, out of vocabulary (OOV) is a big problem while predicting. Some kinds of word segmentation technologies have been proposed (Joulin et al., 2017; ray Su and yi Lee, 2017) to solve these problems.

BPE (Sennrich et al., 2016) has been proposed to handle these problems as well. It shows its powerful effectiveness in many works (Sennrich and Haddow, 2016; Radford et al., 2018; Devlin et al., 2018). However, it is originally designed to handle open-vocabulary problem in machine translation, basing on the intuition that various word classes are translatable via smaller units than words, for instance names (via character copying or transliteration), compounds (via compositional translation), and cognates and loanwords (via phonological and morphological transformations). Thus compared with semantic characteristics, morphological and compounding characteristics are more considered. The unique meanings of some proper

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 163–173
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

nouns are missing while simply applying it to some NLU tasks. Consider the following textual entailment example:

Example 1. t. *Traditionally, the* ***Brahui*** *of the* ***Raisani*** *tribe are in charge of the law and order situation through the Pass area. This tribe is still living in present day* ***Balochistan*** *in Pakistan.*

h. *The* ***Raisani*** *tribe resides in Pakistan.*

$t \rightarrow h$: *entailment*

BPE:

Brahui: bra hu i

Raisani: rai san i

Balochistan: bal o chi stan

In this example, proper nouns play an important role, but they are divided into wordpieces shared with other words. Especially they have common pieces such as "i" and "o". Therefore, inferring from the encoded sequence can be quite difficult.

Apart from this, let us motivate the lack of concern on some key words or phrases of such method. Here is another example:

Example 2. t. *Cairo is now home to* ***some 15 million*** *people - a burgeoning population that produces* ***approximately 10,000 tonnes*** *of rubbish per day, putting an enormous strain on public services...*

h. ***15 million tonnes*** *of rubbish are produced* ***daily*** *in Cairo.*

$t \rightarrow h$: *not_entailment*

In this case, "15 million tonnes" is the key phrase while "15", "million" and "tonnes" also appear in the context. Word-level or subword-level information is obviously not enough.

To alleviate these issues, we propose a reverse mapping bytepair encoding method to integrate prior knowledge into subwords. The prior knowledge mentioned here includes named-entity information, part-of-speech (POS) tags and dependency parsing labels. Our method has two forms and both of them modify the encoding procedure of BPE. In the conventional form, firstly we tag and parse the target sentence which needs to be tagged with NER, POS taggers as well as a dependency parser. After that, we handle the target sentence with the original BPE algorithm. Note that the taggers and the parser work on the word-level while BPE works on the wordpiece-level. Finally, we encode every wordpiece as a combination of linguistic features of its parent word and itself. To avoid over-reliance on the performance of external tools and error propagation, we modify the former as named-entity phrase based reverse mapping, which adds named-entity phrases to the vocabulary during the scanning process of the whole corpus and encodes a wordpiece as the combination of the named-entity phrase where the wordpiece comes from and itself. We evaluate our method on a set of NLU tasks by applying it to GPT. More specifically, we add a weighted layer after the embedding layer to get different weighted combinations of the inputs. We also propose a new model architecture named as the multi-channel separate transformer to employ a training process without parameter-sharing for wordpieces and additional features. We evaluate our approach on two natural language inference tasks (RTE, SciTail), a question answering task (Story Cloze Test) and a classification task (SST-2), showing the benifits of our approach.

2 Related Work

Improving natural language understanding requires better techniques for modeling natural language. There have been many researchers working on better capturing semantic and morphological information of word vectors (Mikolov et al., 2013a,b; Huang et al., 2012; Levy and Goldberg, 2014b; Pennington et al., 2014). Utilizing internal information has been widely studied, and most of these works employed structural information between words and smaller units (Chen et al., 2015; Iacobacci et al., 2015; Bojanowski et al., 2017; Yu et al., 2017; Xu et al., 2018). Another relative research direction is to use external knowledge. The researchers in Microsoft (Song et al., 2011) employed a big and rich probabilistic knowledge-base to machine learning algorithms, and got significant improvement in terms of tweets clustering accuracy. However, such method needs huge human and material resources to build up a high-quality and extremely wide-coverage knowledge base. Recently, a novel language representation model called ERNIE (Zhang et al., 2019) has been proposed. ERNIE introduces knowledge-related tasks in the pre-training process. Besides, it utilizes graph embedding methods to get the embedded representation of entities. Compared with it, our method does not rely upon external knowledge bases and it can be a double-edged sword. In addition, our method provides syntactic information integration and allows principled integration of named-entity information in an easier way.

There is also a class of method, instead of relying a lot on external knowledge, it takes advantage of the linguistic features that exist in natural language. Levy (Levy and Goldberg, 2014a) generalized the skip-gram model to include arbitrary contexts by dependency parsing. The dependency-based embeddings are less topical and exhibit more functional similarity than the original skip-gram embeddings. Multimodal representations of chinese characters (ray Su and yi Lee, 2017) have also been studied, and the research showed the effectiveness of glyph features in some cases. Sennrich (Sennrich and Haddow, 2016) proposed an approach to employ linguistic features for neural machine translation (NMT). These features include lemmas, subword tags, morphological features, POS tags and dependency labels. Different from us, they paid more attention on how to improve NMT from the morphological level. Besides, they did not consider named entities. Another work relevant to us is (Nallapati et al., 2016). They proposed a feature-rich encoder to capture keywords in summarization. Their model employs a word-level vocabulary, and the words are embedded by concatenating all kinds of features including POS, NER tags and discretized TF and IDF values. Different from them, our approach works on the subword level and leverages linguistic features at the semantic level.

There are many kinds of neural networks that can deal with NLU tasks. Recently, pre-trained language models or language representation models have been widely used and these works got significant improvements (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018). Trained on large corpora will inevitably face the big-vocabulary problem and the OOV problem. Thus researchers have proposed several methods to handle them. FastText (Joulin et al., 2017) employed n-grams thus it could predict the zero-shot word embeddings. However, n-gram is an arbitrary method of word segmentation. Sennrich (Sennrich et al., 2016) adapted the original byte pair encoding (BPE) compression algorithm to NMT and got significant success. This method iteratively merges most frequent adjacent characters or character sequences in a word until it can not be done, based on the well learned token rank. Therefore it can encode any word with a pre-learned token vocalulary. We give an illustration as Figure 1, showing the ranking process of BPE. GPT (Radford et al., 2018) applied BPE to its training process and got impressive achievements in a series of NLU tasks. Due to the greedy nature of BPE algorithm, the results produced by BPE are deterministic, resulting in insufficient robustness in machine translation. Kudo (Kudo, 2018) located this problem and proposed a subword regularization method to handle it. He trained the NMT model with multiple subword segmentations generated in a probability manner and got improvements especially on low resource and out-of-domain settings. This study provides an interesting insight, but we argue that it may not bring significant improvement in NLU tasks. In fact, BPE is proposed to model open-vocabulary translation in NMT, which is inconsistent with the goal of some natural language understanding tasks. That motivates us to start our work.

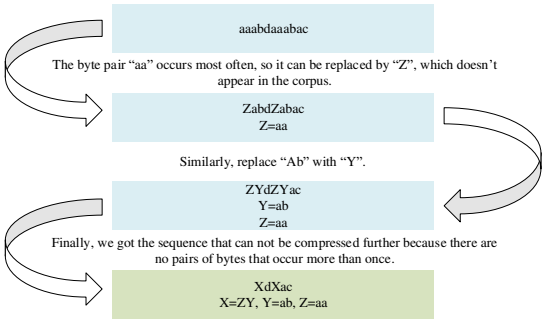


Figure 1: Ranking process of bytepair encoding.

3 Methods

In this section, we introduce our reverse mapping bytepair encoding method and the procedure of applying it to GPT. The reverse mapping bytepair encoding has two forms: label based (LB-BPE) and named-entity phrase based (NPB-BPE). Figure 2 provides a visual illustration. We employ them to the fine-tuning process of GPT by adding a weighted layer after the embedding layer. Besides, we propose a multi-channel separate transformer (MCST) to evaluate the utility of introduced features and give some insights into the semantic capture capabilities of the transformer. The model architecture is shown in Figure 3.

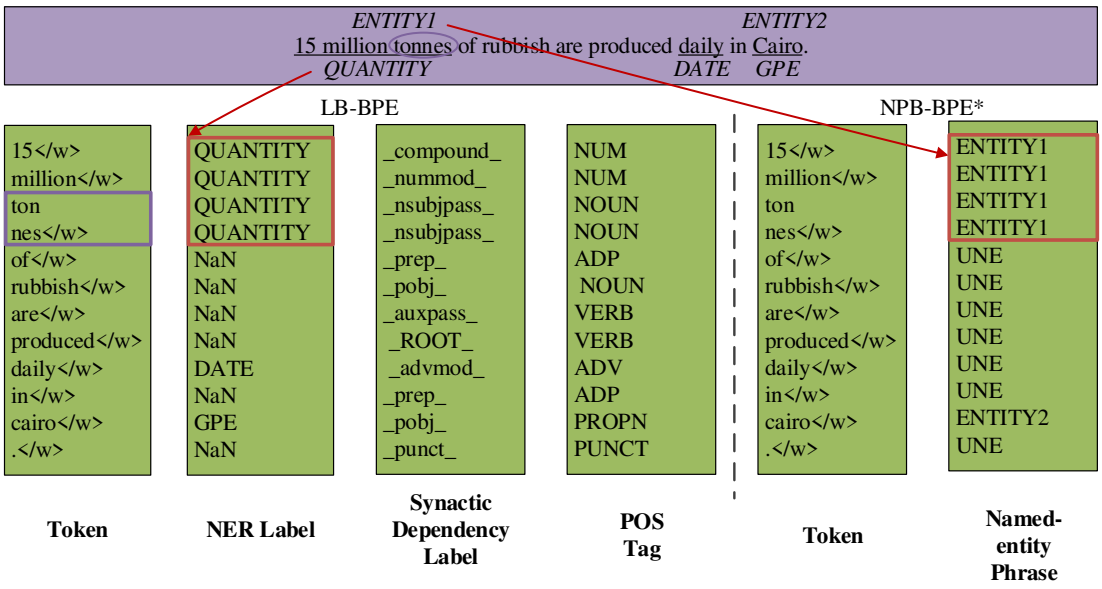


Figure 2: "tonnes" is encoded to be two tokens: "ton", "nes⟨/w⟩". **(Left)** LB-BPE result. Each token consists of four parts: token, NER label, synactic dependency label and POS tag. Words in the original text pass their linguistic features to the tokens generated from them. **(Right)** NPB-BPE* result. Each token consists of two parts: token and corresponding entity.

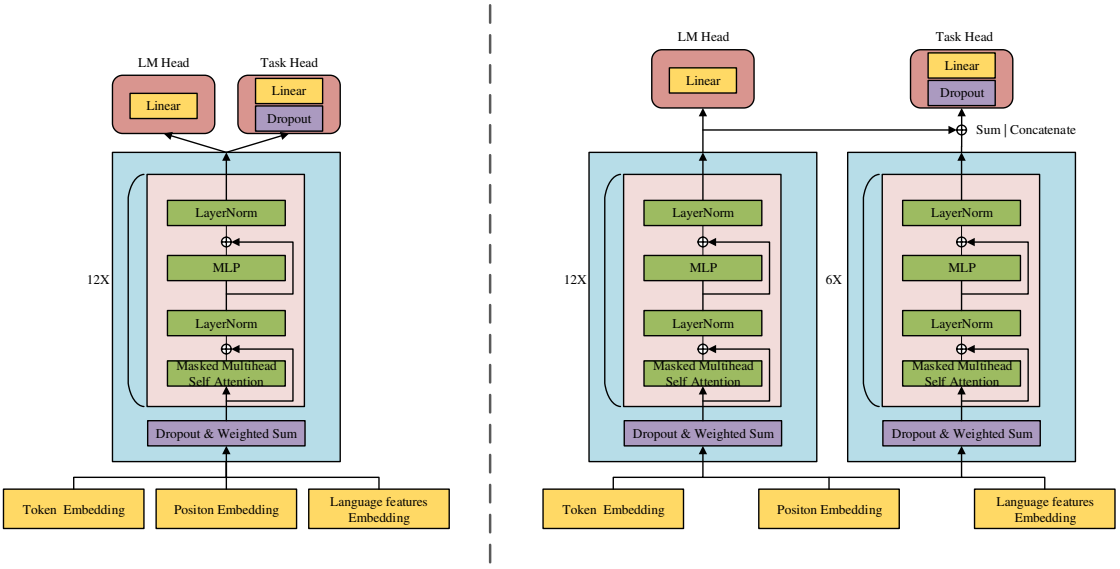


Figure 3: **(Left)** Parameter-sharing training procedure. **(Right)** Multi-channel separate transformer. Language model loss is only generated by the left transformer. The input of the task specific head is the sum or concatenation of both transformers outputs.

3.1 Label Based Reverse Mapping Bytepair Encoding

We extend a token t generated by BPE to four parts $\{t, t_{pos}, t_{ner}, t_{dep}\}$, where t is the same as the original encoded token in BPE, t_{pos} is the POS tag of the word which generates this token, similar with t_{ner} and t_{dep}. Not all words are tagged with named-entity labels, therefore we tag these tokens without NER labels as "NaN". We regard t_{pos}, t_{ner}, t_{dep} as additional features that could not be captured fully by language models. Table 1 shows the amount of features of various types.

Feature type	Amount
NER	20
POS	19
DEP	51

Table 1: Amount of different types of features.

Schemes come from *Spacy Annotation Specifications*[1]. We use *spaCy*[2] to tokenize, tag and parse the datasets during our experiments. We simply use the dependency labels instead of the whole dependency parsing tree because of two reasons: one is it brings complex changes to the input architecture, the other is that we assume the transformer architecture has the ability to establish some kinds of patterns to capure the dependencies between tokens generated by BPE.

3.2 Named-entity Phrase Based Bytepair Encoding

Algorithm 1: Encoding process for NPB-BPE

Input :
Given sentence S;
Pretrained lookup table T for tokens;
Lookup table T_{ne} for named-entity phrases;
NER Scheme Θ := {B,I,O};
Output:
Encoded sequence S';

Current named-entity token array e;
for *each word* $w \in S$ **do**
  Step1. **if** *NER*(w_i) $\in$ {*B,I*} **then**
    Append w_i to e;
    i++;
    Back to Step1;
  end
  else
    e_str := ARR2STR(e)
    if $e_str \notin T_{ne}$ **then**
      Append e_str to T_{ne};
    end
    for $t' \in$ *BPE*(e_str) **do**
      Append (t', e_str) to S';
    end
    Empty e;
    for $t \in$ *BPE*(w) **do**
      Append (t, "_UNE_") to S';
    end
  end
end

As we mentioned in Introduction 1, some key words are low-frequency. Such as a person name "Kevin Federline", it can be encoded as "kevin⟨/w⟩", "feder" and "line⟨/w⟩", and the meaning will be changed. Passing the keyword information to tokens is a simple and feasible approach to solve this problem. There are many ways of defining which words are key words and thus the problem can be defined as following:

Definition 1. Find an algorithm $A, \forall k \in S$, $A(k) = 1$ if k is a key word, else $A(k) = 0$.

where S represents a sentence.

This is a two-category classification problem. In fact, these key words usually perform as people's names, organizations or other types of named entities. Therefore, we choose a NER algorithm[3] as algorithm A and we modify the encoding process of BPE algorithm as algorithm 1 to pass the entity attribute to tokens. By reverse mapping named-entity words or phrases appeared in the corpus back to BPE tokens, key information has a way to be preserved. Some kinds of named entities (ep. DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, etc) contain numbers and they usually express general attributes, such as "22-year-old". We add a switch in practice to control whether processing these words with NPB-BPE or not.

The main difference between LB-BPE and NPB-BPE is that LB-BPE makes ordinary use of additional semantic information that can not be easily captured by statistical language models in a specific language, while NPB-BPE provides a way to share important concepts within a corpus.

3.3 Training Process

GPT is followed by many studies for its excellent generalization performance and we will give a brief introduction to it in the first subsubsection. Due to the expensive cost of pre-training, we reuse the OpenAI pre-trained language model[4] parameters, and we train new embeddings and fine-tune all parameters during the fine-tuning process. Figure 4 shows the modified preprocess procedure. Considering the input is enhanced by different kinds of new features, we propose two different processes for training.

Generative Pre-trained Transformer

The Generative Pre-trained Transformer (Radford et al., 2018), also known as OpenAI GPT or called as GPT, is a language representation model which

[1] https://spacy.io/api/annotation#_title
[2] https://spacy.io/
[3] https://spacy.io/api/entityrecognizer
[4] https://github.com/openai/finetune-transformer-lm

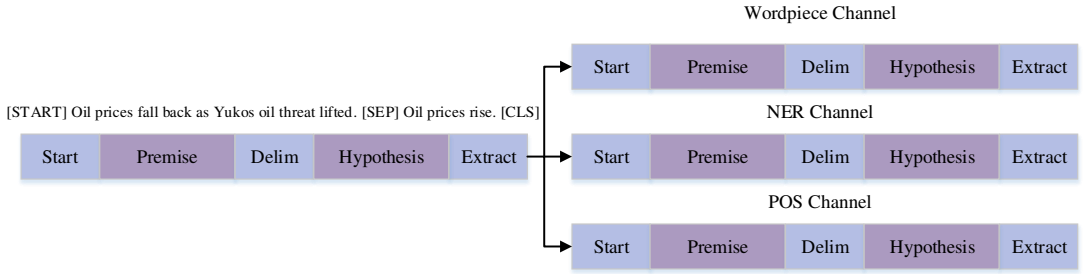


Figure 4: The preprocess procedure of RTE dataset by LB-BPE (NER+POS). Others are similar with this example.

is pre-trained on large corpora and fine-tunes all pre-trained parameters on the downsteam tasks. Its main architechture was originally described in (Vaswani et al., 2017). Unless otherwise stated, the GPT mentioned in this paper refers to a 12-layer decoder-only transformer with masked self-attention heads (768 dimensional states and 12 attention heads). For the position-wise feed-forward networks, we use 3072 dimensional inner states. Other hyperparameters are set as well as the GPT paper (Radford et al., 2018) for comparison.

Sharing Parameters

With this method, we treat additional features like positional encoding. We add these tags to the vocabulary and random initialize their embeddings in the embedding layer, thus the input can be described as following:

$$h_0 = w_{wt}E_t + w_{wp}E_p + \sum_{i \in \mathcal{C}_{lf}} w_{wli}E_{li} \quad (2)$$

where E_t, E_p, E_{li} represent the token embedding matrix, position embedding matrix and linguistic feature embedding matrix, $\mathcal{C}_l$ is the collection of all types of linguistic features and w_{wt}, w_{wp}, w_{wli} are the corresponding weight scalars. The following data flow is the same as GPT (Liu et al., 2018). We use the following objective to minimize:

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda L_1(\mathcal{C}) \quad (3)$$

where $\mathcal{C}$ represents the labeled dataset, $L_2(\mathcal{C})$ is the cross entropy loss of classification got on $\mathcal{C}$, $L_1(\mathcal{C})$ is the cross entropy loss of the language model got on $\mathcal{C}$ and λ is a weight parameter between 0 and 1.

Multi-channel Separate Transformer

GPT uses a multi-layer transformer decoder as the language model, and it learns patterns in the language by simply observing the token-level sequences. In this part, we use LB-BPE to encode sequences and with this method we have multiple input channels. We employ two stand-alone multi-layer transformer decoder to separate the parameters learned on different perspectives. One is the 12-layer pre-trained transformer released by OpenAI GPT, and the other is an entirely new one which has different amount of layers with the former and takes responsibility for linguistic features.

4 Experiments and Analysis

4.1 Setup

We follow the model hyperparameters mentioned in the GPT (Liu et al., 2018) paper. We use learned position embeddings with variable length depending on downstream tasks. We perform a pilot experiment and the result shows that it is almost the best to set the weighted layer as 1.0 for each channel. We use a 16G *ASPEED Graphics Family (rev 30)* card for training.

4.2 Supervised Fine-tuning

Datasets

Story Cloze Test Story Cloze Test (Mostafazadeh et al., 2017) is a new commonsense reasoning framework for evaluating story understanding, story generation, and script learning. This test requires a system to choose the correct ending to a four-sentence story. We choose it as a part of our evaluation corpora because it requires the model to capture rich linguistic phenomena.

RTE and SciTail RTE (Bentivogli et al., 2009) and SciTail (Khot et al., 2018) are language in-

Model	Story Cloze(Acc%)	RTE(Acc%)	SciTail(Acc%)	SST-2(Acc%)
jose.fonollosa's model	87.60	-	-	-
BiLSTM+ELMo+Attn	-	58.9	-	90.4
BigBird	-	-	**93.84**	-
GPT	86.5	56.0	88.3	91.3
NER	**87.39**	**61.6**	**88.90**	91.2
NER+POS	**88.08**	**62.1**	**88.33**	91.1
NER+DEP	**87.55**	**62.0**	**88.99**	90.2
NER+POS+DEP	**87.07**	**60.4**	87.54	**92.1**
POS	84.07	**62.4**	88.15	**92.1**
POS+DEP	86.05	**61.5**	87.63	**91.7**
DEP	86.48	**62.0**	87.82	91.3
NPB-BPE	**87.76**	**58.8**	**88.43**	91.1
NPB-BPE*	**87.39**	**62.3**	87.16	**91.9**

The details of the models for comparison in this table can be found in the leaderboards of the corresponding dataset official websites. Each row in the second block represents a LB-BPE result with different combination of features. NER represents using named-entity labels, POS represents part of speech tags and DEP represents synactic dependency parsing tags. NPB-BPE* represents filtering out named entities within these types (DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL).

Table 2: Reverse mapping bytepair encoding results on Story Cloze Test, RTE, SciTail and SST-2.

ference also called textual entailment tasks which given a text t and a hypothesis h, t entails h, if, typically, a human reading t would infer that h is most likely true. The relation is directional. These two datasets differ from each other in size and domain.

SST-2 SST-2 (Socher et al., 2013) is related to sentiment analysis, containing 56.4k movie reviews and each of them has a binary label.

Evaluation of Label Based Reverse Mapping Bytepair Encoding

We evaluate LB-BPE on these datatsets and conduct multiple controlled trials based on different combinations of linguistic features by employing the parameter-sharing training procedure. Details is described in Table 2. The result shows that incorporating named-entity features usually works well, while utilizing POS tags sometimes can bring significant performance improvement. Besides, it is not a good way to use external features as many as possible, most likely because of the error propagation. However, combinations of named-entity information and others may bring a small boost. Compared with GPT, we achieved an absolute increase of 1.58% on Story Cloze Test, 6.4% on RTE, 0.69% on SciTail and 0.8% on SST-2 at our best.

Evaluation of Named-entity Phrase Based Bytepair Encoding

We evaluate NPB-BPE with the parameter-sharing training procedure. Details are shown in Table 2. Overall, NPB-BPE improves the performances for all datasets while not overly relying on external features. On Story Cloze Test, both kinds of NPB-BPE are not worse than LB-BPE (NER). On RTE, NPB-BPE* nearly achieves the best performance of LB-BPE. Little performance boost is shown on SciTail, probably because sentences in SciTail are more focused on the expressions of concepts and knowledge, and thus key words are more about verbs and nouns rather than named entities. However, as we unexpected, applying NPB-BPE* to SST-2 gains a 0.6% absolute increase. We also collect the statistics of named entities on these datasets. Details is shown in Table 3. We observe that there is a positive correlation between named entities percentage and performance improvement.

Evaluation of Different Training Processes

The parameters in GPT are all the same for all wordpiece embeddings in the vocabulary. However, linguistic features contain different levels of information compared with words and wordpieces. Thus we evaluate two different training processes mentioned in Section 3.3. As for MCST,

Dataset	Amount	Percentage(%)	Top10(non-numeric)
Story Cloze	2354	7.3	(“One day”, 241), (“John”, 196), (“Amy”, 195), (“Bob”, 189), (“Joe”, 156), (“Tom”, 152), (“Tim”, 148), (“Sam”, 114), (“Gina”, 101), (“Jim”, 95)
RTE	15548	21.9	(“U.S.”, 294), (“Iraq”, 262), (“US”, 243), (“the United States”, 180), (“China”, 166), (“today”, 150), (“France”, 148), (“Monday”, 144), (“American”, 144), (“Bush”, 142)
SciTail	7631	4.8	(“Earth”, 670), (“earth”, 632), (“Sun”, 358), (“Mercury”, 262), (“Oxygen”, 139), (“Hydrogen”, 134), (“Venus”, 103), (“Jupiter”, 101), (“Weight”, 101), (“Mars”, 92)
SST-2	728	1.6	(“american”, 141), (“years”, 128), (“summer”, 122), (“the year”, 119), (“year”, 90), (“today”, 87), (“2002”, 60), (“this year”, 54), (“two hours”, 53), (“saturday”, 51)

Table 3: Statistics of named entities on Story Cloze Test, RTE, SciTail, SST-2.

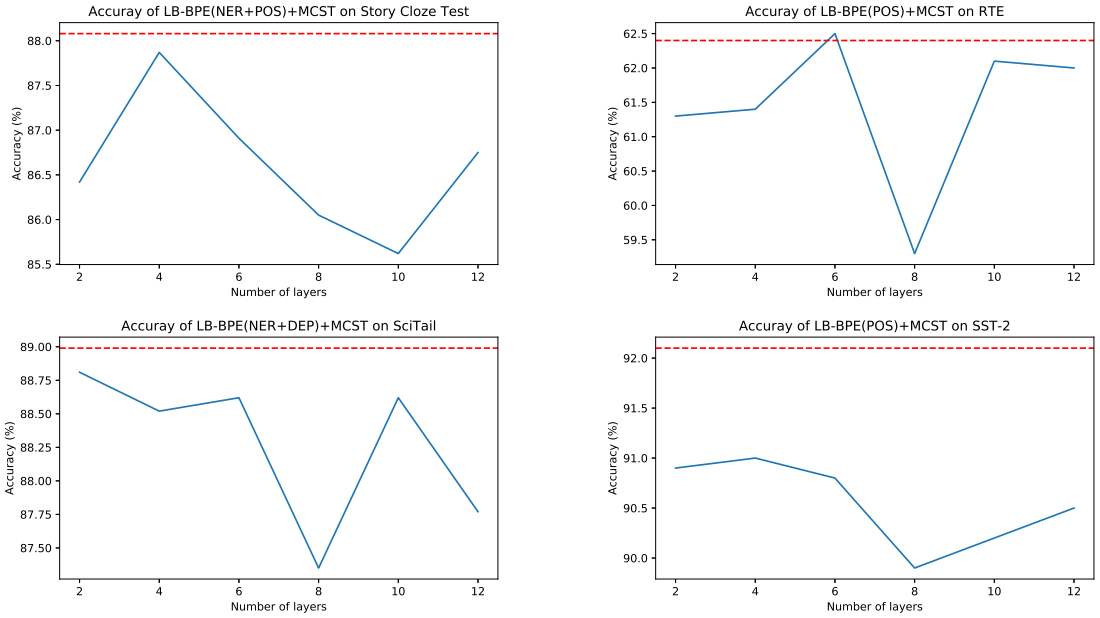


Figure 5: Accuracy of LB-BPE+MCST on all datasets. The red dotted line in each subgraph represents the best result of LB-BPE+GPT got on that dataset.

we perform a series of experiments based on different amount of blocks in the new transformer. As shown in Figure 5, employing non-parameter-sharing training process will generally reduce the performance. We infer that it is probably due to the lack of pre-training procedure for newly introduced features. However, MCST predicts labels based on both transformers while the accuracy rates do not drop too much, which suggests that the newly introduced linguistic features work at least to some extent and the transformer architecture can capture some potential semantic information from them independently. Besides, MCSTs trained with 2, 4, 6 layers in the new transformer perform best in our experiments, and they gain similar or even better results compared with the parameter-sharing training process, which means training newly introduced features doesn't need as many parameters as the original transformer. The main reason is that linguistic features act at a higher level compared with words and wordpieces.

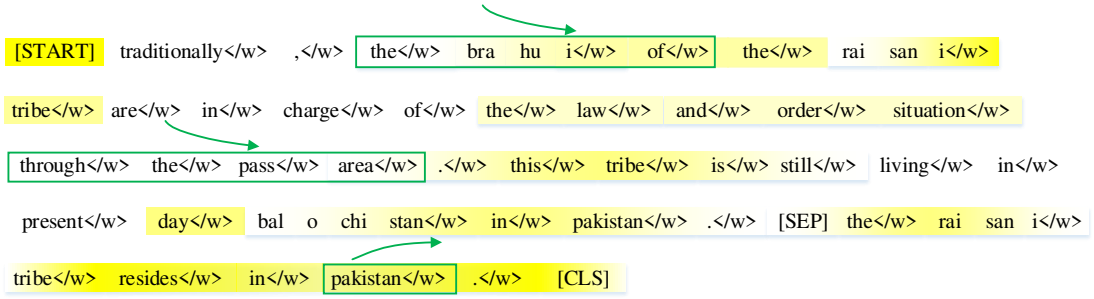


Figure 6: Attention weights of a textual entailment case in RTE. Yellow parts represent the attention weights of our approach. Green arrows and boxes represent the weight changes compared with GPT.

4.3 Case Study

We compare the attention weights of GPT and our LB-BPE (NER+POS) approach in a textual intailment example as shown in Figure 6. GPT labels it as *not_entailment* while our approach labels it as *entailment* which is the right answer. The attention weights come from the same attention head in the last transformer block. Changes in some parts lead to the correct answer.

5 Conclusion and Future Work

In this paper, we introduce a simple approach called reverse mapping bytepair encoding to improve natural language understanding based on pre-trained language models. The reverse mapping bytepair encoding has two forms: One is label based and the other is named-entity phrase based. Both forms introduce extra information to the tokens generated by BPE while the former ordinarily employs linguistic features, and the latter provides a way to share concepts through named-entity phrases. In addition, in the second form, we summarize the problem we are trying to solve into a two-category problem that judges whether a word is a key word. By applying them to the fine-tuning process of GPT, we gain about 1-6% improvement on downsteam NLU tasks. We also propose a new model architecture named as MCST and experiments based on it shows its effectiveness in some cases. Besides, the experimental results show that linguistic features usually perform at a higher level compared with words and wordpieces. It is quite easy to apply our method to the existing language models.

There could be several directions to be explored for future works. Language models have many forms, we only test our approach on GPT, a follow up direction is finding if it is generic enough. In this paper, we don't employ pre-training for linguistic features, it might be better by doing this. There are several researches focusing on incorporating knowledge into systems to improve their performances, thus we are looking forward to finding a smooth way to utilize named entities with prior knowledge or knowledge graphs.

Acknowledgments

This work is supported by National Key Research and Development Program of China under Grant No. 2017YFB0803003 and No. 2016YFB0801302. We also thank the anonymous reviewers and the area chairs for their valuable comments and suggestions that help improve the quality of this manuscript.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo. Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huanbo Luan. 2015. Joint learning of character and word embeddings. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Yangqiu Song, Haixun Wang, Zhongyuan Wang, Hongsong Li, and Weizhu Chen. 2011. Short text conceptualization using a probabilistic knowledgebase. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Tzu ray Su and Hung yi Lee. 2017. Learning chinese word representations from glyphs of characters. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yang Xu, Jiawei Liu, Wei Yang, and Liusheng Huang. 2018. Incorporating latent meanings of morphological compositions to enhance word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 2017. Joint embeddings of chinese words, characters, and fine-grained subcharacter components. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Made for Each Other: Broad-coverage Semantic Structures Meet Preposition Supersenses

Jakob Prange **Nathan Schneider**
Georgetown University
jakob@cs.georgetown.edu
nathan.schneider@georgetown.edu

Omri Abend
The Hebrew University of Jerusalem
oabend@cs.huji.ac.il

Abstract

Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013) is a typologically-informed, broad-coverage semantic annotation scheme that describes coarse-grained predicate-argument structure but currently lacks semantic roles. We argue that lexicon-free annotation of the semantic roles marked by prepositions, as formulated by Schneider et al. (2018), is complementary and suitable for integration within UCCA. We show empirically for English that the schemes, though annotated independently, are compatible and can be combined in a single semantic graph. A comparison of several approaches to parsing the integrated representation lays the groundwork for future research on this task.

1 Introduction

A common thread in many approaches to meaning representation is the idea that abstract structures can describe semantic invariants that hold across paraphrasing or translation: for example, semantic dependency relations capturing predicate-argument structures or other types of semantic relations that can be annotated within sentences (e.g., Böhmová et al., 2003; Oepen et al., 2015; Banarescu et al., 2013). These annotation schemes can be distinguished by various design principles such as language-specificity; the level of granularity of meaning elements; the reliance on morphosyntactic criteria to define the units of semantic annotation; the extent to which human annotators specify semantics from scratch; and many others (Abend and Rappoport, 2017).

In this work, we seize an opportunity to unite two previously unrelated—yet complementary—meaning representations in NLP. On the one hand, Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013) provides

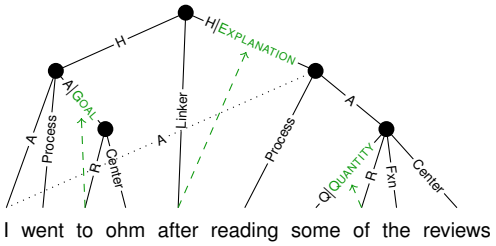


Figure 1: Semantic parse illustrating the integrated representation proposed here. Solid edges are the UCCA parse's *primary edges*, and the dotted edge is a *remote edge*. Dashed arrows show how SNACS labels (green small caps) have been mapped onto edges of the UCCA structure from the prepositions on which they were originally annotated. The following UCCA categories are abbreviated: A = Participant, R = Relator, H = Parallel scene, Q = Quantifier, Fxn = Function.

a skeletal structure of semantic units and relations, with typologically-based criteria for marking predicate-argument structures, based on "Basic Linguistic Theory", an established framework for typological description (Dixon, 2010/2012). On the other hand, a recent approach to annotation of English prepositions and possessives (SNACS; Schneider et al., 2018) provides an inventory of labels that characterize semantic relations. UCCA and SNACS follow similar design principles: they are both *language-neutral*, with general-purpose coarse-grained labels rather than lexically-specific senses or roles; and they are both designed for *direct* semantic annotation, without requiring a syntactic parse as a foundation. The philosophy is that these properties will facilitate annotation in many languages and domains that may lack detailed lexicons. But UCCA makes only the most rudimentary role distinctions, while SNACS annotations thus far have not made explicit which elements are being brought into a semantic relation (§2).

We propose a design that achieves the best of both worlds, as illustrated for an English example in figure 1. Taking advantage of an English corpus

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 174–185
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

that has been separately annotated for both UCCA and SNACS, as well as dependency syntax, we show that the SNACS role labels can be automatically integrated within UCCA structures over a range of syntactic constructions (§3). Then, we use this corpus to test pipelined, multitask, and joint approaches to parsing the integrated representation (§4). Our findings (§5) set the stage for future English parsers as well as multi- and cross-lingual extensions. §6 situates this work in the broader landscape of computational meaning representations.

Our main contributions are:

- a typologically-oriented broad-coverage linguistic **representation** that captures predicate-argument structure and semantic roles, without reference to any lexicon;
- a procedure to **integrate** UCCA and (token-level) SNACS annotations for particular sentences, mapping the SNACS labels to the appropriate edge in the UCCA structure, by which we create an integrated gold standard; and
- initial results for the **integrated parsing task**, comparing several alternatives that couple the learning/prediction of UCCA and SNACS in various ways. We find that optimizing for the two objectives jointly works best.

Data and code from these experiments are open-sourced to facilitate future work on this task.[1]

2 Background

To better understand the distinctions that we expect to be captured by such a framework, consider the following examples:

(1) a. Her picture was on the wall.
b. Her speech was on security.

Despite parallel syntax and overlapping vocabulary, the sentences above vary in numerous aspects of meaning:

- The NPs *her picture* and *the wall* denote entities that stand in a certain locative relation to each other, as signaled by the preposition *on*.
- In contrast, the relation between *her speech* (which is an event, not an entity) and *security* is a different one, TOPIC, despite being signaled by the same preposition.

This is made explicit in the German translations of these sentences:

(2) a. Ihr Bild hing **an** der Wand .
Her picture hung **at** the wall .
b. Ihre Rede war **über** Sicherheit .
Her speech was **over** security .

In addition, the possessive pronoun *her* (*ihr*/*ihre*) signals a prototypical POSSESSION relation in (1a)/(2a), but the core role of AGENT in (1b)/(2b).

As we can see, the natural lexical choices for expressing the LOCATION relation between the picture and the wall and the TOPIC relation in German have diverging literal translations to English. Thus, the empirical study of cross-linguistic commonalities and differences between form and meaning calls for a common metalanguage to describe the relations between mentioned events and entities, as marked by case and adpositions.

Our approach to such a representation consists of utilizing two existing semantic representations. UCCA (§2.1) captures the structure of predicate-argument and head-modifier relations at a high level, crucially distinguishing units that evoke a **scene** (event or state) from other units. SNACS (§2.2) disambiguates semantic roles as signaled by adpositions and possessives, but only directly annotates a function word, without formalizing the semantic relation that it mediates.[2] Both of these schemas have the guiding principle to be language-independent, eschewing a lexicon and defining a closed inventory of semantic categories.

2.1 UCCA

UCCA is a semantic annotation scheme rooted in typological and cognitive linguistic theory. It aims to represent the main semantic phenomena in the text, abstracting away from syntactic forms. UCCA's foundational layer, which is the only layer annotated over text so far,[3] reflects a coarse-grained level of semantics that has been shown to be preserved remarkably well across translations (Sulem et al., 2015). It has also been successfully used for improving text simplification (Sulem et al., 2018b),

[1] Integrated data: https://github.com/jakpra/ucca-streusle; parser code: https://github.com/jakpra/tupa; the integration routine and evaluation scripts are being released as part of the UCCA PyPI package and under https://github.com/jakpra/ucca.

[2] Note that mapping between syntactic and semantic relations varies by construction: in *She spoke on security*, the semantic head of the relation between *spoke* and *security* corresponds to the syntactic head (the verb), whereas in *Her speech was on security*, UD treats *security* as the syntactic head (§3).

[3] Prange et al. (2019) have proposed and piloted a coreference layer that sits above the foundational layer.

as well as to the evaluation of a number of text-to-text generation tasks (Birch et al., 2016; Sulem et al., 2018a; Choshen and Abend, 2018).

Formally, UCCA structures are directed acyclic graphs over **units** (nodes covering a subset of tokens). Atomic units are the leaves of the graph: individual tokens or unanalyzable MWEs. Nonterminal units represent larger semantic constituents, such as scenes and compositional participants/modifiers. The example in figure 1 has 5 nonterminal units. Each unit (save for the root) has a single incoming **primary** edge, and may also have incoming reentrant **remote** edges to express shared argumenthood. The primary edges of a UCCA structure thus form a tree, which along with the remote edges, forms a DAG.[4]

Edges are labeled with one or more **categories** indicating a kind of semantic relationship. The small set of categories includes State and Process for static or dynamic scene predicates, respectively; Participant, Time, and Adverbial for dependents of scenes; Center for the head of a non-scene unit (usually an entity); and Elaborator and Quantity for modifiers of entities. Scenes can be semantic dependents (Participant of another scene, Elaborator of a non-scene). Multiple scenes at the same level are called Parallel Scenes, and connectives between them are Linkers.

UCCA makes a distinction between different functions of prepositions, the most common cases of which are: (1) phrasal verbs (e.g., "give **up**"), annotated as internally unanalyzable; (2) linkers between scenes; e.g., in figure 1, "after" links the going to ohm scene, and the reading scene); (3) main relations in scenes (e.g., "The apple tree is **in** the garden"); and (4) case markers within a scene or a participant, or *Relators* in UCCA terms (e.g., in figure 1, "to" and "of" are such markers).

However, apart from distinguishing temporal modifiers, the UCCA scheme does not provide any **semantic role** information: thus the analyses of "the dark wizard defeated by Gandalf" and "the dark wizard's defeat of Gandalf" are nearly isomorphic—obliterating the distinction between agents and patients in the semantics—though the grammatical encoding of the noun phrases in question (subject, by-PP, possessive, of-PP) leaves no ambiguity about the intended roles to a human reader.

[4]There is also the capability to annotate implicit units, but these are ignored in the standard evaluation and we do not address them here.

2.2 SNACS

SNACS is an inventory of 50 roles/relations used to disambiguate adpositions and possessives in multiple languages, including English (Schneider et al., 2018, 2019), Mandarin Chinese (Zhu et al., 2019), and to a lesser extent, Korean, Hindi, and Hebrew (Hwang et al., 2017). Many of the SNACS labels, such as AGENT, THEME, and TOPIC, are derived from VerbNet's (Kipper et al., 2008) core roles of predicates. (Others, such as QUANTITY and WHOLE, are for entity modification.) But unlike VerbNet, FrameNet (Fillmore and Baker, 2009), and PropBank (Palmer et al., 2005), SNACS does not require a language-specific predicate lexicon (hence Schneider et al. (2018) use the term "supersenses", which we adopt in the remainder of this paper)—and is therefore compatible with UCCA's design principle of crosslinguistic applicability.[5]

Currently, SNACS labels are applied directly to lexical items, without marking up underlying structure on either the subword (morphological) or the sentence-structure level.

3 Automatically Integrating Semantic Roles with UCCA

With the benefit of a jointly annotated corpus, we examined the data and determined that the proper placement of adpositional semantic role labels is fairly deterministic given certain syntactic patterns and their structural counterparts in UCCA. Here we present a rule-based method for automatically integrating token-based semantic role annotations from SNACS into an UCCA graph as edge refinements. We use these rules to construct a gold standard for analysis and parsing of the integrated representation. The rules we use, though empirically grounded in the English Web Treebank corpus (Bies et al., 2012), make no specific assumptions about language or lexicon, as they solely depend on UCCA, SNACS, and Universal Dependencies (UD; Nivre et al., 2016) annotation, all of which are frameworks designed to be cross-linguistically applicable. Thus, we expect the rules could be adapted to other languages with only minor changes if the underlying annotations are applied consistently, though this will require testing in future work.

[5]SNACS also annotates the **function** of a preposition token—its lexical semantics which may be distinct from its semantic role (Hwang et al., 2017). Only scene roles are taken into account in the present analysis.

	scene	non-scene
verb	I *went* [**to** ohm]	*Quit* [**with** the overstatements] !
noun	Wonderful *service* [**for** large group]	Cheapest *drinks* [**in** Keene]
	[10 minutes] **of** *paperwork*	[No amount] **of** *sugar* and milk can mask it .

Table 1: Syntactic and semantic dimensions of canonical adpositional phrase constructions. The adposition is **bolded**, the semantic head is *italicized*, and the semantic dependent is [bracketed]. Rows indicate whether the semantic head is nominal or verbal, while columns differentiate between scene-evoking and non-scene-evoking heads. Scene-evokers are underlined.

3.1 Data

We use the STREUSLE 4.0 corpus (Schneider and Smith, 2015; Schneider et al., 2018), which covers the reviews section from the English_EWT treebank of UD 2.3, and lexical semantic annotations for the same text.[6] The same corpus has been annotated with UCCA by Hershcovich et al. (2019a). We use the standard train/dev/test split for this dataset (table 2). §3.3 shows the distribution of linguistic phenomena at issue here.

3.2 Procedure

Given an UCCA graph with annotated terminals, the integration routine projects a SNACS annotation of a token onto the appropriate edge representing a semantic relation. This is illustrated by dashed arrows in figure 1. The procedure starts with a single terminal node, traversing the graph upwards until it finds an edge that satisfies the criteria given by the rules. The rules concern canonical prepositional phrase modifiers, plus a variety of syntactically or otherwise anomalous constructions, such as copulas and adverbs.

3.2.1 Canonical PPs

The adpositional constructions annotated in STREUSLE can be adnominal or adverbial modifiers and arguments, where both the nouns and verbs that are being elaborated on can evoke either scenes or non-scene units in UCCA (table 1). First, we take a look at expressions marked with Relators in UCCA, which generally correspond to prototypical syntactic PPs.

Modifiers of scenes. In general, if the adposition marks a modifier of a scene—i.e., the adposition is the first or last terminal in the modifier unit's yield—the supersense should refine the role of that dependent. The adposition's parent unit is refined by the supersense ("to ohm" in figure 1 and table 1; "for large group" in table 1).

	train	dev	test	total
sentences	2,723	554	535	3,812
tokens	44,804	5,394	5,381	55,579
SNACS-annotated	4,522	453	480	5,455
successful integ.	4,435	447	473	5,355
matches synt. obj	3,924	403	438	4,765

Table 2: Quantitative analysis of adpositional constructions in the corpus.

Modifiers of non-scenes. Where the adposition relates a modifying (elaborating or quantifying) unit to a non-scene unit, the supersense refines the modifying unit. If the adposition is the first or last terminal in a non-Center unit, that unit gets refined ("in Keene" in table 1). This includes the case when the adposition marks the predicate evoking the scene of which the syntactic governor is a modifier ("with the overstatements": "Quit" is treated as an aspectual modifier of the "overstatements" event).

For partitive constructions like "the top **of** the mountain", both the syntactic governor and object of the adposition are marked as UCCA Centers, indicating that they are on a *semantically equal* level—neither one is clearly only modifying or being modified by the other. In this case, the supersense refines the syntactic object of the adposition.

Quantities. Another special case is where the adposition is labeled as QUANTITY, in which case the unit for its syntactic governor[7] ("10 minutes" in table 1) receives the refinement: e.g., "[some] **of** the reviews" in figure 1; in table 1, "[10 minutes] **of** paperwork" and "[no amount] **of** sugar" (the bracketed expression is the QUANTITY).

3.2.2 Non-canonical phenomena

For other, less prototypical constructions involving SNACS-annotated expressions, such as copulas, linked parallel scenes, possessive pronouns, and idiomatic PPs, we have additional rules, summarized in supplementary material (appendix A).

[6] UD: https://github.com/UniversalDependencies/UD_English-EWT; however, as described in §5.1, we use automatic dependency parses in all experiments, to emphasize generalizability.

[7] We determine the head noun of the syntactic governor and object using a script released together with the STREUSLE corpus: https://github.com/nert-nlp/streusle/blob/master/govobj.py. Since semantic UCCA units do not always align with syntactic phrases, we choose the UCCA unit containing the head token of the syntactic governor or object in its yield.

	train	dev	test	total
total primary edges	54,204	6,628	6,623	67,455
total remote edges	2,881	349	387	3,617
refined	4,473	449	479	5,401
≥ 1 edge refined	38	2	6	46
remote edges	33	2	5	40
canonical	2,468	242	270	2,980
scene mod	2,124	219	254	2,597
non-scene mod	344	23	16	383
non-canonical	2,005	207	209	2,421
predication	167	19	23	209
linkage	461	54	41	556
intransitive adp.	261	19	23	303
scn-mod nscn-mod	189 72	14 5	20 3	223 80
approximator	14	0	1	15
possessive pron.	897	81	95	1,073
scn-mod nscn-mod	774 123	72 9	87 8	933 140
infinitival	66	18	11	95
scn-mod nscn-mod	15 51	0 18	1 10	16 79
PP idiom	139	16	15	170

Table 3: Refined UCCA edges by construction type, according to our heuristic. (The non-canonical subcategories are mutually exclusive.)

3.3 Quantitative Analysis

We run the integration routine on our dataset and report statistics in tables 2 and 3. The heuristic rules have a coverage of 98% – 99% (row 'successful integ.' divided by row 'SNACS-annotated' in table 2). 88.5% (train) – 92.6% (test) of refined units contain the syntactic complement[8] (table 2), indicating that while syntax may often give a good approximation to the semantic relations marked by adpositions, a direct mapping from syntactic into semantic structure is not always trivial.

In table 3, we see that among the canonically adpositional SNACS targets, the vast majority mark scene modifiers (including participants). The various non-canonical targets modify both scenes and non-scenes, except for linkages and predications, which naturally only operate at the scene-level, and approximators, which only elaborate on non-scenes. Similar to canonical adpositional constructions, possessive pronouns and intransitive adpositions have a tendency to modify scenes rather than non-scenes. Those infinitivals that are not inter-scene Linkers (which are counted under *linkage*) are mostly non-scene modifiers.

Remote edges are only rarely affected by the SNACS integration, so we exclude them from evaluation in §5.

[8]We use the term 'syntactic complement' to include head tokens of prepositional objects and subordinate clauses, as determined by the script mentioned above. If no prepositional object is given in the STREUSLE corpus, we consider the adposition itself and only count towards this metric if the refined edge is its incoming preterminal edge.

3.4 Difficult Cases & Limitations

Our heuristics are based solely on universal semantic and syntactic annotations, with no assumptions about the grammar or lexicon of any specific language. However, there are some limitations to the rules that deserve discussion. Most importantly, as many rules are highly sensitive to UCCA structure and categories, errors or inconsistencies in human and automatic UCCA annotation are likely to throw the system off. This can be mitigated with strict constraints and careful reviewing during manual annotation, but cannot be fully avoided when applying the rules to automatically generated UCCA.

Multiword expressions (MWEs) are another source of difficulty. Both UCCA and STREUSLE mark idiomatic MWEs, but follow slightly different guidelines. The heuristic rules actually recover from most MWE misalignments; however, constructions that are MWEs in UCCA and contain multiple SNACS targets, such as *as-as* comparatives, are not fully resolved by our heuristic, as we cannot assign individual edge refinements for the adpositions' competing supersenses, given that we start traversing the UCCA graph from the shared preterminal node.

4 Models

We hypothesize that our combined (lexical and structural) semantic representation is not only linguistically plausible, but also useful in practice, as the annotations on both levels should be informative predictors for each other. That is, we expect that knowing what semantic role is signaled by an adposition informs the underlying semantic structure of the sentence, and vice versa.

In order to test this hypothesis, we use our annotated corpus to parse into the integrated representation. We consider several different ways of orchestrating the prediction of the foundational UCCA structure and the prediction of SNACS roles: modeling SNACS and UCCA in (i) a pipeline, (ii) a multitask setup with shared parameters, (iii) a single joint model.

4.1 Baseline: TUPA

We choose the neural transition-based graph parser TUPA (Hershcovich et al., 2017, 2018) as a strong baseline for UCCA parsing. It was the official baseline in the recent SemEval shared task on UCCA parsing (Hershcovich et al., 2019b).

TUPA's transition system is defined to address

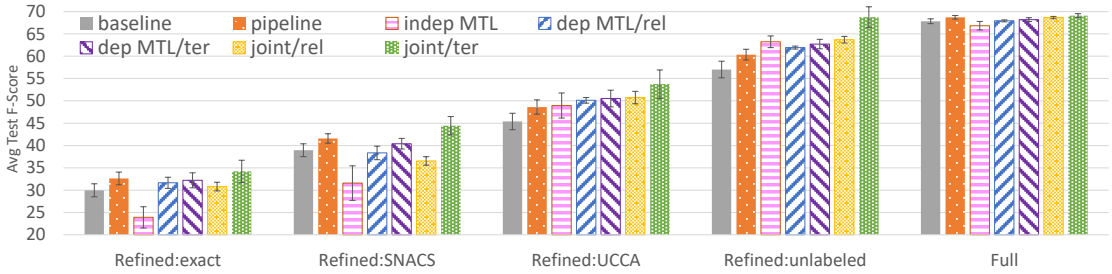


Figure 2: Average F1-score on the test set over 5 random restarts with error bars indicating standard deviation. *ter* stands for terminal-level and *rel* for relation-level SNACS refinement (prediction or features).

the different formal structural phenomena exhibited by UCCA structures, notably reentrancies and discontiguous units. There are transitions for creating nonterminal nodes, and for attaching terminal and nonterminal nodes to a primary or remote (reentrant) parent with an UCCA category label on the edge. The transition system is general enough to be able to tackle parsing into a variety of formalisms, including SDP (Oepen et al., 2015) and a simplified form of AMR (Banarescu et al., 2013); Hershcovich et al. (2018) take advantage of this flexibility in their multitask learning framework.

TUPA's learning architecture largely follows that of Kiperwasser and Goldberg (2016). It encodes the parser's configuration (buffer, stack and intermediate graph structure) using BiLSTMs, and predicts the next transition using an MLP, stacked on top of them. Token-based features, including POS tags, dependency parses, as well as NER and orthographic features, are embedded in the BiLSTM. Another set of features, taking into account the partially constructed graph and previously predicted transition types, is fed into the MLP directly.

4.2 Pipeline

We extend TUPA by providing the SNACS label as a feature on the adposition token.[9] This is added in preprocessing in the same way as the syntactic features listed above (including the BiLSTM encoding). At testing time, we obtain SNACS labels for automatically identified targets from the SVM model of Schneider et al. (2018).

4.3 Multitask

Hershcovich et al. (2018) showed that UCCA parsing performance can be improved with multitask learning (MTL; Caruana, 1997) on several semantic and syntactic parsing tasks. We examine whether alternately optimizing two objectives, one for UCCA and one for SNACS, leads to mutually favorable biases via shared parameters. There are multiple ways the two tasks can be orchestrated:

Independent MTL. This is the multitask learning (MTL) setup from Hershcovich et al. (2018), where separate transition classifiers are trained on different tasks simultaneously, sharing and mutually updating the BiLSTM encoding.[10] We consider as auxiliary tasks (a) SNACS scene role classification and (b) the decision of which UCCA unit is refined by a SNACS-annotated token. We encode these tasks as parsing tasks analogous to UCCA parsing as follows: for each training item in (a), we create a graph consisting of a root and up to 4 children: the syntactic governor (if available), the preposition token, the syntactic object (if available)—all of which have dummy edge labels—as well as a dummy terminal carrying the SNACS supersense. For each training item in (b), we consider the full UCCA structure, but the edge labels are simply boolean values indicating whether an edge is refined or not.

We also train a separate model with SNACS classification as the primary task and UCCA parsing and SNACS integration as auxiliary tasks, whose predictions are integrated in postprocessing for the combined evaluation (table 4), and which is evaluated independently in table 5.

Dependent MTL. Here we train the SNACS task in direct interaction with the UCCA parsing task.

[9] We report here only results for the setting in which a supersense is added as a feature of the *preposition* token. We also experimented with using it as a feature of the syntactic object token—which often, but not always, heads the semantic object (cf. table 2)—but got similar or worse results.

[10] Note that our setup differs from that of Hershcovich et al. (2018) in two key points: In contrast to the auxiliary tasks used in the aforementioned work, SNACS prediction as formulated by Schneider et al. (2018) is not a structured, but a (per-token) classification task (however, as described above, we transform it into an artificially structured task to make it conform with the input format expected by TUPA). Furthermore, we are interested in both UCCA and SNACS performance, expecting both tasks to benefit from each other's complementary semantic content.

system		Refined: exact			Refined: SNACS			Refined: UCCA			Refined: unlabeled			Full		
setup	ref	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
BL	–	30.2	29.7	30.0	39.3	38.7	38.9	45.7	45.1	45.4	57.4	56.6	57.0	68.2	67.6	67.8
(oracle SNACS)		45.4	45.1	45.2	62.7	62.3	62.5	48.9	48.6	48.7	62.7	62.3	62.5	69.5	68.9	69.2
pipeline	ter	32.9	32.4	32.6	42.0	41.3	41.6	49.1	48.2	48.6	60.9	59.9	60.4	68.8	68.6	68.7
(oracle SNACS)		53.5	53.2	53.3	70.4	70.0	70.2	57.0	56.7	56.9	70.4	70.0	70.2	71.0	70.7	70.8
indep MTL	ter	26.0	22.2	23.9	34.3	29.3	31.6	53.1	45.4	49.0	68.7	58.6	63.2	67.1	66.6	66.8
dep MTL	ter	34.4	30.3	32.2	43.1	38.0	40.4	53.9	47.5	50.5	66.9	59.1	62.7	68.4	68.1	68.2
	rel	32.7	30.6	31.6	39.6	37.2	38.3	51.8	48.6	50.1	64.0	60.1	61.9	68.3	67.6	67.9
joint	ter	**34.5**	**34.3**	**34.2**	**44.6**	**44.3**	**44.4**	53.9	**53.5**	**53.7**	69.0	**68.5**	**68.7**	**69.5**	**68.7**	**69.1**
	rel	34.0	28.1	30.8	40.4	33.4	36.5	**56.1**	46.4	50.7	**70.3**	58.2	63.7	69.1	68.3	68.7

Table 4: Experimental results, averaged over 5 random restarts. The baseline system (BL) for UCCA is TUPA version 1.3.9 without any modifications, retrained on our data. For the sake of generalizability and consistency with our own preprocessing, we use system-predicted SNACS categories from the auto-id/auto-syntax setting from (Schneider et al., 2018) in the BL and pipeline setups. Results where the system has access to gold SNACS annotations on adposition tokens are shown in small font.

We enhance TUPA with a separate MLP that, given an edge, classifies its supersense refinement (a null category can be chosen to indicate an unrefined edge). This network is run after each edge-creating transition. Its input features are the same as for the transition classifier, including the BiLSTM encoding. Since the two classifiers alternate in making forward passes and updating the shared BiLSTM, they indirectly contribute to each other's input. Here we have an option of where in the UCCA structure to initially predict the supersense label. In the **terminal-level (ter)** setting, we predict supersense refinements only on preterminal edges, and then apply the integration rules (§3) as postprocessing. In the **relation-level (rel)** setting, we parse directly into the integrated representation. To do this, we preprocess the training data with our integration routine. However, during parsing, there is no explicit restriction that supersense-refined edges must have an adposition token in their yield—thus the model could, in theory, learn to predict adequate role supersenses even when it is not signaled by a lexical marker (though it will get penalized for that in our current evaluation).

4.4 Joint

Finally, we train a single classifier on the integrated data, concatenating UCCA and SNACS categories, to predict parsing transitions using the new compound categories. We revisit the terminal-level and the relation-level settings introduced in §4.3.

5 Experiments

5.1 Experimental Setup

Preprocessing. We follow Hershcovich et al. (2018) in obtaining automatic POS and NE tags, as well as syntactic dependency relations using SpaCy 2.0, and pretrained word vectors from fastText.[11] For all setups that use or predict SNACS supersenses, we include the gold standard scene role categories for pre-identified targets from the STREUSLE 4.0 corpus in our training and development data. In the test data we identify adposition targets using the heuristics introduced in Schneider et al. (2018). For the joint prediction setup (§4.3), we also include the head terminal of the syntactic governor and object for each adposition as features, using the same heuristics as in §3.

Architecture and hyperparameters. For classifying the next transition, TUPA uses a multi-layer perceptron (MLP) with 2 hidden layers and a softmax output layer, on top of a 2-layer BiLSTM (Kiperwasser and Goldberg, 2016). Building on previous work, we train for 100 epochs, using the stochastic gradient descent optimizer for the first 50, and AMS-grad (Reddi et al., 2018) for the remaining 50 epochs,[12] and apply early stopping post-hoc by keeping the model with the highest performance on the dev set as the final model.

Evaluation. For our main evaluation in §5.2, we compare our systems along five new metrics: a **full structure** score which evaluates precision and recall of all units and requires both the UCCA and SNACS categories to be correct, where applicable; and **refined UCCA**, **SNACS**, **exact**, and **unlabeled** scores which only consider SNACS-refined units. Here, the integrated representation obtained via the rule-based integration (§3.2) serves as the ground truth. We also report the standard labeled and unlabeled UCCA scores.[13] In addition, for sys-

[11] https://spacy.io/; https://fasttext.cc/

[12] Except for the independent MTL setting, where we stop training after the first 50 epochs.

[13] All of the above metrics are F-scores over the edges, as in Hershcovich et al. (2017).

system		UCCA labeled			UCCA unlabeled			SNACS		
setup	ref	P	R	F	P	R	F	P	R	F
BL	–	72.5 ±0.6	71.9 ±.4	72.2 ±0.4	88.5 ±0.3	87.6 ±.5	88.0 ±.3	58.5	58.3	58.4
pipeline	ter	72.4 ±0.3	**72.0** ±.6	72.2 ±0.4	88.3 ±0.2	**87.8** ±.6	88.1 ±.4	–	–	–
(oracle SNACS)		73.0 ±0.4	72.6 ±.7	72.8 ±0.5	88.8 ±0.2	88.3 ±.7	88.5 ±.4	–	–	–
indep MTL	ter	71.0 ±1.2	70.2 ±.9	70.6 ±1.0	87.9 ±1.0	86.9 ±.8	87.3 ±.7	48.2 ±5.6	41.4 ±4.8	44.6 ±5.2
dep MTL	ter	71.8 ±0.4	71.3 ±.2	71.5 ±0.3	88.1 ±0.4	87.6 ±.2	87.8 ±.2	60.1 ±1.8	53.3 ±2.0	56.5 ±1.9
	rel	71.8 ±0.3	71.0 ±.1	71.4 ±0.1	88.0 ±0.3	87.0 ±.4	87.5 ±.3	–	–	–
joint	ter	**72.8** ±0.3	71.9 ±.4	**72.3** ±0.3	**88.7** ±0.3	87.6 ±.4	**88.2** ±.3	**60.5** ±3.0	**60.3** ±3.4	**60.4** ±3.2
	rel	72.5 ±0.4	71.5 ±.2	72.0 ±0.3	88.5 ±0.2	87.2 ±.3	87.8 ±.2	–	–	–

Table 5: Results on the respective tasks of UCCA parsing and token-level SNACS prediction, averaged over 5 random restarts, with standard deviation reported next to each average. The baseline system (BL) for UCCA is TUPA version 1.3.9 without any modifications, retrained on our data. The SNACS baseline system is the SVM classifier of Schneider et al. (2018).

tems which predict a terminal-level SNACS label (before it is mapped to a higher relation in post-processing), we compare SNACS disambiguation performance against (Schneider et al., 2018) in §5.3.

5.2 Integrated parsing results

Our MTL and joint systems outperform the baseline and a feature pipeline on refined UCCA units (figure 2 and table 4). The main benefit from considering UCCA and SNACS together in training is that the parser is better at recovering the (unlabeled) structure of units that should receive a SNACS relation in the integrated representation. This is illustrated in figure 3. This trend is confirmed in the precision and recall of UCCA units that have a gold SNACS token in their yield (unlabeled F-score: *BL* = 93.1, *dep-MTL/ter* = 95.2, *indep-MTL* = 96.0, see table 7 in the supplementary material). To the extent that these units are syntactic constituents (see table 2), this suggests that multitask learning with syntactic auxiliary tasks (Swayamdipta et al., 2018; Hershcovich et al., 2018) might be particularly beneficial for SNACS-augmented UCCA parsing. The feature pipeline is competitive, but noisy features from a previous classification step limit its performance on refined units. The upper bounds given by the oracle setting indicate that SNACS features are generally beneficial. *Indep-MTL* and systems that parse directly into the relation-refined representation struggle with predicting the correct SNACS refinements—and thus also exact UCCA+SNACS combinations—while the *joint/ter* model is consistently the most accurate.

However, there is little effect on overall labeled and unlabeled UCCA scores (table 5). Predicting SNACS simultaneously or interactively with UCCA (*joint/rel* and *dep-MTL*) apparently makes the parsing task harder. Note that particularly in

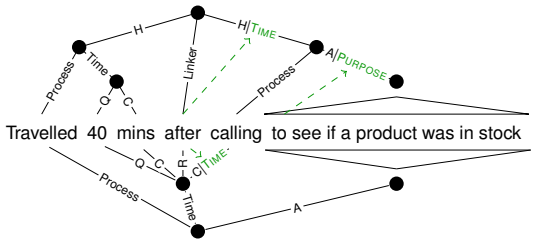


Figure 3: Simplified sample output. The *joint/ter* system (top) generates the intended scene structure and PURPOSE modifier attachment. The *BL* system (bottom) does not, and misses the PURPOSE role altogether.

setup	ref	# params	
BL	–	78.9M	
pipeline	ter	78.9M	
indep MTL	ter	UCCA 82.8M	SNACS 81.8M
dep MTL	ter	79.4M	
	rel	79.4M	
joint	ter	79.2M	
	rel	79.2M	

Table 6: Number of parameters of each model.

the *dep-MTL* setting, erroneous decisions in one task could negatively affect the other.

5.3 Token-based SNACS prediction results

Since some of our systems predict SNACS labels at the terminal level, they are directly comparable to previous work on SNACS classification. We compare against the auto-id/auto-syntax baseline from Schneider et al. (2018) in table 5.[14] Both the *dep MTL* and *joint* systems outperform the baseline in precision; and the *joint* system also in recall, leading to the overall best performance. The *indep-MTL* system does not reach baseline performance.

5.4 Model capacity

We examine whether the differences in performance can really be attributed to the linguistic in-

[14] Due to the diverging guidelines on multiword units in UCCA (§3.4), we ignore MWE boundaries.

formation in our data or merely to more powerful models by inspecting the number of each model's parameters (table 6). While we observe some variance in model capacity, we consider these to be minor differences. An exception is the *independent MTL* setup, which consists of two independent models, each dedicated to a specific task. However, this does not seem to give it an advantage in terms of final performance. The baseline has the fewest parameters, and the overall best condition, *joint/ter*, is neither the smallest nor the largest model, suggesting that the particular linguistic signals and the method of using them have a genuine effect on performance.

6 Related Work

The benefits of integrating lexical analysis and sentence semantic or syntactic structure have been pursued by a vast body of work over the years. Compositional approaches to the syntax-semantics interface, such as CCG (Steedman, 2000) and HPSG (Pollard and Sag, 1994), usually integrate the lexicon at the leaves of the syntactic parse, but propagate grammatically-relevant features up the tree. A different approach is taken by OntoNotes (Hovy et al., 2006), which consists of a number of separate, albeit linked tiers, including syntactic and argument structure, but also the lexical tiers of word senses and coreference.

Role semantics frequently features in structured semantic schemes. Some approaches, such as PropBank and AMR (Palmer et al., 2005; Banarescu et al., 2013), follow a lexical approach. The Prague Dependency Treebank tectogrammatical layer (Böhmová et al., 2003) uses a few lexicon-free roles, but their semantics is determined by virtue of their linkage to a lexicalized valency lexicon. Universal Decompositional Semantics (White et al., 2016) instead defines roles as a bundle of lexicon-free features, elicited by crowdsourcing.

The specific inventory for preposition/possessive relations that we use is SNACS, but there is a wider history of disambiguation of these items, especially in English: disambiguation systems have been described for possessives (Moldovan et al., 2004; Badulescu and Moldovan, 2009; Tratz and Hovy, 2013), prepositions with lexicalized sense definitions (e.g., Litkowski and Hargraves, 2007; Tratz and Hovy, 2011), and prepositions with coarse-grained classes (O'Hara and Wiebe, 2003, 2009; Srikumar and Roth, 2013; Gonen and Goldberg, 2016). Such disambiguation has also been investigated in tandem with semantic role labeling and parsing (Dahlmeier et al., 2009; Srikumar and Roth, 2011; Gong et al., 2018). Preliminary work suggests that SNACS may be applicable to subjects and objects, not just PPs, and thus in the future this framework could be extended to all UCCA participants (Shalev et al., 2019).

State-of-the-art results on UCCA parsing and SNACS disambiguation are described in contemporaneous work by Jiang et al. (2019); Liu et al. (2019), who achieve substantial gains using the ELMo and BERT contextualized word embeddings (Peters et al., 2018; Devlin et al., 2019). This is an orthogonal direction to the one we pursue here, and combining the two is left to future work.

7 Conclusion

We have introduced a new representation combining UCCA semantic structures and SNACS adpositional semantic roles; automatically merged existing annotations to create a gold standard; and experimented with several alternatives for parsing the integrated representation. Our results show that models profit from having access to both structural and lexical semantic information, confirming our hypothesis that UCCA and SNACS are complementary and compatible.

Based on preliminary results from a German corpus, we conjecture that this approach is applicable to other languages with no or only minimal changes—a direction we will explore further in future work. In addition, we plan to investigate the utility of the enhanced representation for downstream tasks involving meaning-preserving linguistic variation.

Acknowledgments

We would like to thank Daniel Hershcovich and Adi Shalev for letting us use their code and helping fix bugs; Sean MacAvaney and Vivek Srikumar for assistance with computing resources; as well as three anonymous reviewers for their helpful comments and suggestions. This research was supported in part by NSF award IIS-1812778 and grant 2016375 from the United States–Israel Binational Science Foundation (BSF), Jerusalem, Israel.

References

Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proc. of ACL*, pages 228–238, Sofia, Bulgaria.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proc. of ACL*, pages 77–89, Vancouver, Canada.

Adriana Badulescu and Dan Moldovan. 2009. A Semantic Scattering model for the automatic interpretation of English genitives. *Natural Language Engineering*, 15(2):215–239.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank. Technical Report LDC2012T13, Linguistic Data Consortium, Philadelphia, PA.

Alexandra Birch, Omri Abend, Ondřej Bojar, and Barry Haddow. 2016. HUME: Human UCCA-based evaluation of machine translation. In *Proc. of EMNLP*, pages 1264–1274, Austin, Texas.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague Dependency Treebank: A three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*, Text, Speech and Language Technology, pages 103–127. Springer Netherlands, Dordrecht.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28(1):41–75.

Leshem Choshen and Omri Abend. 2018. Reference-less measure of faithfulness for grammatical error correction. In *Proc. of NAACL-HLT*, pages 124–129, New Orleans, Louisiana.

Daniel Dahlmeier, Hwee Tou Ng, and Tanja Schultz. 2009. Joint learning of preposition senses and semantic roles of prepositional phrases. In *Proc. of EMNLP*, pages 450–458, Suntec, Singapore.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT*.

Robert M. W. Dixon. 2010/2012. *Basic Linguistic Theory*. Oxford University Press.

Charles J. Fillmore and Collin Baker. 2009. A frames approach to semantic analysis. In Bernd Heine and Heiko Narrog, editors, *The Oxford Handbook of Linguistic Analysis*, pages 791–816. Oxford University Press, Oxford, UK.

Hila Gonen and Yoav Goldberg. 2016. Semi supervised preposition-sense disambiguation using multilingual data. In *Proc. of COLING*, pages 2718–2729, Osaka, Japan.

Hongyu Gong, Suma Bhat, and Pramod Viswanath. 2018. Embedding syntax and semantics of prepositions via tensor decomposition. In *Proc. of NAACL-HLT*, pages 896–906, New Orleans, Louisiana.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proc. of ACL*, pages 1127–1138, Vancouver, Canada.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proc. of ACL*, pages 373–385, Melbourne, Australia.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2019a. Content differences in syntactic and semantic representation. In *Proc. of NAACL-HLT*, pages 478–488, Minneapolis, Minnesota.

Daniel Hershcovich, Zohar Aizenbud, Leshem Choshen, Elior Sulem, Ari Rappoport, and Omri Abend. 2019b. SemEval-2019 Task 1: Cross-lingual Semantic Parsing with UCCA. In *Proc. of SemEval*, pages 1–10, Minneapolis, Minnesota, USA.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *Proc. of HLT-NAACL*, pages 57–60, New York City, USA.

Jena D. Hwang, Archna Bhatia, Na-Rae Han, Tim O'Gorman, Vivek Srikumar, and Nathan Schneider. 2017. Double trouble: the problem of construal in semantic annotation of adpositions. In *Proc. of *SEM*, pages 178–188, Vancouver, Canada.

Wei Jiang, Zhenghua Li, Yu Zhang, and Min Zhang. 2019. HLT@SUDA at SemEval-2019 Task 1: UCCA graph parsing as constituent tree parsing. In *Proc. of SemEval*, pages 11–15, Minneapolis, Minnesota, USA.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Ken Litkowski and Orin Hargraves. 2007. SemEval-2007 Task 06: Word-Sense Disambiguation of Prepositions. In *Proc. of SemEval*, pages 24–29, Prague, Czech Republic.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proc. of NAACL-HLT*, pages 1073–1094, Minneapolis, Minnesota.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju. 2004. Models for the semantic classification of noun phrases. In *HLT-NAACL 2004: Workshop on Computational Lexical Semantics*, pages 60–67, Boston, Massachusetts, USA.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: a multilingual treebank collection. In *Proc. of LREC*, pages 1659–1666, Portorož, Slovenia.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajič, and Zdeňka Urešová. 2015. SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing. In *Proc. of SemEval*, pages 915–926, Denver, Colorado.

Tom O'Hara and Janyce Wiebe. 2003. Preposition semantic classification via Treebank and FrameNet. In *Proc. of CoNLL*, pages 79–86, Edmonton, Canada.

Tom O'Hara and Janyce Wiebe. 2009. Exploiting semantic role resources for preposition disambiguation. *Computational Linguistics*, 35(2):151–184.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: an annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL-HLT*, pages 2227–2237, New Orleans, Louisiana.

Carl Pollard and Ivan A. Sag. 1994. *Head-driven Phrase Structure Grammar*. University of Chicago Press.

Jakob Prange, Nathan Schneider, and Omri Abend. 2019. Semantically constrained multilayer annotation: the case of coreference. In *Proc. of the First International Workshop on Designing Meaning Representations*, pages 164–176, Florence, Italy.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of Adam and beyond. In *Proc. of ICLR*, Vancouver, British Columbia, Canada.

Nathan Schneider, Jena D. Hwang, Archna Bhatia, Vivek Srikumar, Na-Rae Han, Tim O'Gorman, Sarah R. Moeller, Omri Abend, Adi Shalev, Austin Blodgett, and Jakob Prange. 2019. Adposition and Case Supersenses v2.3: Guidelines for English. *arXiv:1704.02134v4 [cs]*.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Jakob Prange, Austin Blodgett, Sarah R. Moeller, Aviram Stern, Adi Bitan, and Omri Abend. 2018. Comprehensive supersense disambiguation of English prepositions and possessives. In *Proc. of ACL*, pages 185–196, Melbourne, Australia.

Nathan Schneider and Noah A. Smith. 2015. A corpus and model integrating multiword expressions and supersenses. In *Proc. of NAACL-HLT*, pages 1537–1547, Denver, Colorado.

Adi Shalev, Jena D. Hwang, Nathan Schneider, Vivek Srikumar, Omri Abend, and Ari Rappoport. 2019. Preparing SNACS for subjects and objects. In *Proc. of the First International Workshop on Designing Meaning Representations*, pages 141–147, Florence, Italy.

Vivek Srikumar and Dan Roth. 2011. A joint model for extended semantic role labeling. In *Proc. of EMNLP*, pages 129–139, Edinburgh, Scotland, UK.

Vivek Srikumar and Dan Roth. 2013. Modeling semantic relations expressed by prepositions. *Transactions of the Association for Computational Linguistics*, 1:231–242.

Mark Steedman. 2000. *The Syntatic Process*. MIT Press, Cambridge, MA.

Elior Sulem, Omri Abend, and Ari Rappoport. 2015. Conceptual annotations preserve structure across translations: a French-English case study. In *Proc. of the 1st Workshop on Semantics-Driven Statistical Machine Translation (S2MT 2015)*, pages 11–22, Beijing, China.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018a. Semantic structural evaluation for text simplification. In *Proc. of NAACL-HLT*, pages 685–696, New Orleans, Louisiana.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018b. Simple and effective text simplification using semantic and neural methods. In *Proc. of ACL*, pages 162–173, Melbourne, Australia.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. 2018. Syntactic scaffolds for semantic structures. In *Proc. of EMNLP*, pages 3772–3782, Brussels, Belgium.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proc. of EMNLP*, pages 1257–1268, Edinburgh, Scotland, UK.

Stephen Tratz and Eduard Hovy. 2013. Automatic interpretation of the English possessive. In *Proc. of ACL*, pages 372–381, Sofia, Bulgaria.

Aaron Steven White, Drew Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal Decompositional Semantics on Universal Dependencies. In *Proc. of EMNLP*, pages 1713–1723, Austin, Texas, USA.

Yilun Zhu, Yang Liu, Siyao Peng, Austin Blodgett, Yushi Zhao, and Nathan Schneider. 2019. Adpositional supersenses for Mandarin Chinese. In *Proc. of SCiL*, volume 2, pages 334–337, New York, NY, USA.

Generating Timelines by Modeling Semantic Change

Guy D. Rosin[1], Kira Radinsky[1,2]
[1]Technion – Israel Institute of Technology, Haifa, Israel
[2]eBay Research, Israel
{guyrosin,kirar}@cs.technion.ac.il

Abstract

Though languages can evolve slowly, they can also react strongly to dramatic world events. By studying the connection between words and events, it is possible to identify which events change our vocabulary and in what way. In this work, we tackle the task of creating timelines—records of historical 'turning points', represented by either words or events, to understand the dynamics of a target word. Our approach identifies these points by leveraging both static and time-varying word embeddings to measure the influence of words and events. In addition to quantifying changes, we show how our technique can help isolate semantic changes. Our qualitative and quantitative evaluations show that we are able to capture this semantic change and event influence.

1 Introduction

Languages respond to world events in many ways. New words, phrases, and named entities are created, new senses may develop, and valences may change. Various approaches support the study of historical linguistics (e.g., comparative linguistics, etymology, etc.). In this work, we focus on a specific process for tracking the progression of meaning over time in sense, semantics, and in relation to other words and concepts. By leveraging changing relationships in temporal corpora, we demonstrate a way of 'embedding' words and world events. Observing changes in this embedding allows us to construct timelines that support the study of evolving languages.

The timeline of scientific and technical discoveries, for example, can drive the emergence of new word senses as these discoveries are 'named'. Take the word "cell" which evolved from its 12^{th} century meaning (a small room or chamber) to a new sense in the 17^{th} century (a basic unit of an organism) to the 19^{th} century meaning (an electric battery) and most recently to a shorthand for a mobile phone[1]. Critically, the *dominant senses* of a word vary over time as some meanings become less commonly used while others gain in popularity. This dynamic need not be driven only by the addition of certain senses. The prevalence of a hyponym, for example, may also drive a change in the ranking of senses. The word 'disaster' may call to mind very different things depending on the latest *type* of disaster. Thus, the word may evoke 'nuclear disaster' in a reader in 2011 (e.g., driven by the *Fukushima* incident). However, in 2012 the 'storm' sense may be more salient (e.g., driven by *Superstorm Sandy*).

Evolution of senses is but one way a language can evolve. Broader semantic changes can also occur. For example, the valence of the word may move or even flip (e.g., *terrific* or *bully*). Of particular interest to us are those changes that are more immediate and precipitated by key world events. For example, a war may lead certain terms to take on a negative connotation as a country or people become the 'enemy'. Large collections of text from a given period can capture all of these language changes as reflected by evolving context. By mining this text, our goal is to support the study of evolving languages.

Etymological studies allow us to understand the origin of words and changes in meaning (Alinei, 1995). This work produces not only an accounting of change but also an explanation of the social, scientific, or other world events that drive language shifts. Conventional production of etymological analysis often requires a detailed and laborious manual close-reading of historical texts (Geeraerts et al., 1997). By applying computational methods, our focus is on detecting semantic changes of words and events and producing possi-

[1] https://www.etymonline.com/word/cell

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 186–195
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

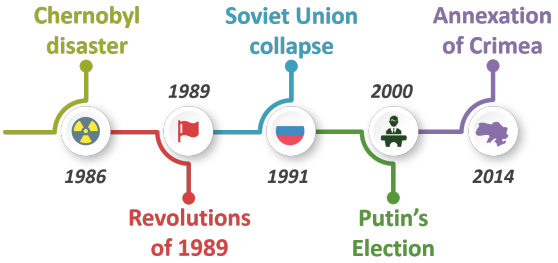


Figure 1: A timeline generated by our framework for *Russia*.

ble explanations from real-world drivers.

Problem Definition: In this work, we study the problem of timeline generation for a word or phrase. We use the term 'word' throughout the paper for convenience. Given a timeline of a word, a researcher should be able to understand the word's dynamics, i.e., the changes the word underwent over time. A timeline is defined as a sequence of time points and their descriptors (i.e., explanations of the changes the word underwent at that time point). A good timeline is such that enables the researcher to gain a better understanding of the word and its history (Althoff et al., 2015). It contains time points of significant changes, with relevant explanations of the changes, and with a minimal number of missing or redundant information.

We define several building blocks for constructing timelines. The first is the identification of time points during which the target word underwent significant semantic change (we refer to those as *Turning Points*, see Section 4). Second, we consider identifying associated descriptors of those changes. These descriptors are associated with a word's change at a particular time and can serve to explain its dynamics.

We experiment with two types of descriptors. The first involves *words* associated with the target word or affected by it (Section 5). The above 'cell' and 'disaster' examples can serve as examples of timelines with word descriptors. The second type is *events* (Section 6). One can explain changes the target word underwent based on significant world events. As an example, consider the timeline generated by our framework for the word "Russia" (Figure 1).

To identify events that are strongly associated with the change, we utilize time-varying language embeddings on both static snapshots (e.g., Wikipedia) and historical texts (35 years of the *New York Times*), allowing us to capture both syntactic and semantic variation of words (Section 3). We develop a mechanism for simultaneously embedding words and events in the same space (Section 6.1). We present several methods to leverage those embeddings for key historical events detection by evaluating the distance between words and events (Sections 6.2, 6.3).

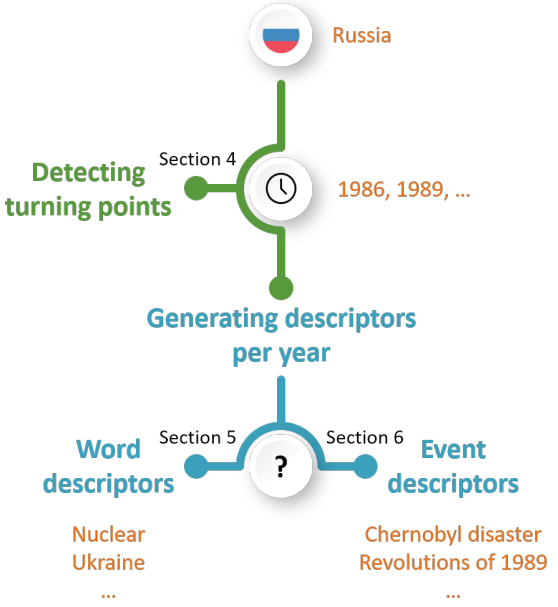


Figure 2: Flow diagram of timeline generation. The two basic building blocks are detecting turning points and generating descriptors, where the descriptors can be either words or events.

Figure 2 presents the flow of the paper through an example. Consider the word 'Russia'. First, we identify its turning points (Section 4), and then generate descriptors – either words (Section 5) or events (Section 6) – to construct its timeline. We contribute several algorithms for identifying significant events leveraging various types of embeddings, including a supervised learning approach.[2]

2 Related Work

Semantic Change: Most work on language evolution has focused on identifying semantic drifts and word meaning changes (see Kutuzov et al. (2018) for a recent survey). Various approaches have pursued the task of detecting changes in word meaning (Sagi et al., 2009; Mitra et al., 2014; Wijaya and Yeniterzi, 2011; Mihalcea and Nastase, 2012; Popescu and Strapparava, 2013; Jatowt and Duh, 2014; Kenter et al., 2015; Hamilton et al., 2016;

[2]Code and data available at https://github.com/guyrosin/generating_timelines

Azarbonyad et al., 2017). Specific approaches include: dynamic embedding models using a probabilistic Bayesian version of Word2Vec (Bamler and Mandt, 2017), pointwise mutual information (PMI) (Yao et al., 2018), and exponential family embeddings (Rudolph and Blei, 2018). Relatedness over time between words has also been studied. Radinsky et al. (2012) showed that words that co-occur in history have a stronger relation, Rosin et al. (2017) introduced the supervised task of temporal semantic relatedness, and Orlikowski et al. (2018) studied diachronic analogies. In our work, we focus on the world events behind the semantic changes—and isolate those events that co-occur with significant language change.

Change Detection for Semantic Shift Analysis: Detecting major changes involves detecting continuous peaks in time series. Kulkarni et al. (2015) and Basile et al. (2016) offered a mean shift model, and Rosin et al. (2017) used a threshold-based method for this task. We utilize the latter approach as it is simpler and more computationally effective.

Timeline Generation: Past work focused on generating timelines by leveraging information retrieval methods. Examples include the use of Facebook data (Graus et al., 2013), Twitter (Li and Cardie, 2014), and Wikipedia (Tuan et al., 2011) to generate context-aware timelines by ranking related entities by co-occurrence with the main timeline entity. Althoff et al. (2015) created timelines by mining a knowledge base, based on submodular optimization and web-co-occurrence statistics. Shahaf and Guestrin (2010) generated a chain of events connecting two news articles. Our work differs from the prior work in several ways. First, we consider the semantic changes a word undergoes and detect the events that influenced them. We study several word embeddings to measure change and relatedness between words and events over time and construct a timeline.

3 Event and Temporal Word Embedding

We consider several methods to represent events and words. These are used to identify semantic changes and generate timeline descriptors. To capture both words and events in the same space we consider *global embeddings*, which are created upon the English Wikipedia (see Section 7.1).

We also utilize the work of Rosin et al. (2017) for *temporal word embeddings*. Timeline construction requires modeling changes in words and events over time. When looking at a specific word, we wish to focus on its relevant meaning at a particular time. Thus, the *temporal word embeddings* are created using data from a large temporal corpus. Specifically, we leverage the New York Times (NYT) archive. The embeddings are generated for every time period (i.e., year) and enable us to investigate how words meanings and relatedness between words change over time (see Section 7.1).

Finally, in order to compare vectors of the same word in different, independently-trained, vector space models, we align every pair of models using Orthogonal Procrustes (Hamilton et al., 2016).

We use the following notations throughout the paper:

Notation 3.1. v_w is the vector representation of a word w.

Notation 3.2. v_w^t is the vector representation of a word w during time t.

Notation 3.3. $NN_k(w)$ is the set of k-nearest neighbors (kNN) of a word w.

Notation 3.4. $NN_k^t(w)$ is the set of k-nearest neighbors (kNN) of a word w during time t.

Notation 3.5. cos is cosine similarity, which we use as a similarity function between embeddings.

4 Timeline Turning Points

A timeline is composed of time points that identify the changes a word underwent. We refer to those as *Turning Points*, and experiment with several methods to identify them. Formally, let w be a target word, and t be a time point. Each method approximates the probability $d_t(w)$ of t to be a turning point of w. The turning points are then selected by performing peak detection (Rosin et al., 2017) on the series of $d_t(w)$ for every t. We experiment with two methodologies for turning point detection, leveraging the embeddings we introduce in Sections 3, 6.1:

(1) Neighborhood: Changes in the neighborhood of a word over time can be used to capture semantic changes of the word. This method measures the difference between the similar words sets of w between two consecutive years. Formally:

$$d_t(w) = 1 - \frac{\left|NN_k^t(w) \cap NN_k^{t-1}(w)\right|}{k} \quad (1)$$

where $NN_k^t(w)$ is the set of k-nearest neighbors (kNN) of w during time t.

(2) EmbeddingSimilarity: Employing word embeddings, we can also look at the change in the embedding vectors of w:

$$d_t(w) = 1 - cos\left(v_w^t, v_w^{t-1}\right) \quad (2)$$

5 Word Descriptors

The second building block of a timeline is *explaining* what triggered the semantic changes. We refer to such explanations as *descriptors*. One can explain semantic changes with words—significant associated words or terms that correlate to the target word's meaning drift. Letting w be the target word, and t be a time point, we are looking for words that became closer to w in vector space during t. Specifically, we look at the nearest neighbors of w at times t and $t-1$, and denote the set of descriptors by D_t:

$$D_t(w) = NN_k^t(w) \setminus NN_k^{t-1}(w) \quad (3)$$

For example, using this method for 'Russia' results in *Soviet Union* and *Soviet* for 1989, when the Soviet Union was dissolving, and *Ukraine*, *Kyrgyzstan*, and *Latvia* for 1990, when these countries attempted to gain independence from the Union.

6 Event Descriptors

As an alternative for word descriptors, we consider event descriptors. To generate these, given a target word w, our task is to identify its change in time t by a set of significant events E that likely affected w. In this section, we first describe the embeddings we use (Section 6.1) and then present several methods for detecting significant events (Sections 6.2, 6.3).

6.1 Projected Embedding for Events

Temporal word embeddings (Section 3) are created for every time period. They enable us to investigate how word meaning and relatedness between words change over time. However, as it may take some time until an event's name is determined and referred to in newspapers, the paper's text may not have meaningful embeddings for those events. For example, the name "World War I" was used only after WWII started. As a result, we are not able to compare events and words.

To address this problem, we leverage the global embeddings (Section 3). Since Wikipedia articles typically contain balanced descriptions of events, they can be a proper basis for event embeddings. Recall that the way these embeddings are created enables creating a common latent space for *both* words and concepts (and specifically, events), allowing us to compare both at the same time. Our solution involves projecting the global model on each temporal one. This way, we create a joint vector space for words and events, which represents a specific time period. We refer to these embeddings as "*Projected Embeddings*".

We assume that most words' meanings do not change over time and learn a transformation of one embedding space onto another, minimizing the distance between pairs of points. Let us define $W_{wiki} \in \mathbb{R}^{|V_{wiki}| \times d_{wiki}}$ as the matrix of embeddings learned from Wikipedia, where d_{wiki} is the embedding size and $|V_{wiki}|$ is the vocabulary size of Wikipedia. Similarly, $W_{nyt}^{(t)} \in \mathbb{R}^{|V_{nyt}^{(t)}| \times d_{nyt}^{(t)}}$ is the matrix of embeddings learned from the NYT at time t, where $d_{nyt}^{(t)}$ is the embedding size and $|V_{nyt}^{(t)}|$ is the vocabulary size of the NYT at time t. We seek a matrix $W^{(t)} \in \mathbb{R}^{|V_{wiki}| \times d_{nyt}^{(t)}}$ that will contain the transformation of W_{wiki} to $W_{nyt}^{(t)}$ for time t. By making an additional simplifying assumption that the vector spaces are equivalent under a linear transformation, we are able to find $W^{(t)}$ by optimizing the following linear regression model:

$$\arg\min_T \sum_{w_i \in V_{wiki} \cap V_{nyt}^{(t)}} \left\| W_{wiki}(w_i)T - W_{nyt}^{(t)}(w_i) \right\|_2^2 \quad (4)$$

where $T \in \mathbb{R}^{d_{wiki} \times d_{nyt}^{(t)}}$. We then obtain the projected matrix:

$$W^{(t)} = W_{wiki}\widehat{T}^{(t)} \quad (5)$$

where $\widehat{T}^{(t)}$ is the result of Equation 4. Similar methods were used in the field of temporal semantics to align embeddings of different time periods to a unified coordinate system (Szymanski, 2017; Kulkarni et al., 2015).

6.2 Similarity-Based Event Detection

We hypothesize that the events closest in vector space to a word should be the most significant to its timeline. We experiment with two score functions that are based on semantic similarity. The descriptors are chosen as the top-scoring events.

ByWord: Given a target word w, we look for its closest events in a specific time. We define a score function of an event e: $score(e) = cos(v_w, v_e)$ where cos is cosine similarity, and v_w and v_e are the respective embeddings of w and e.

ByKNN: We wish to extend the "impact circle" of the event, as events sometimes do not affect a word directly but through other words. We look for events that are closest not only to the target word but also to its neighbors. We calculate the following score for every possible event e:

$$score(e) = avg(\{cos(v_n, v_e) : n \in \{w\} \cup NN_k(w)\}) \quad (6)$$

6.3 Supervised Event Detection

The previous methods discuss only the semantic similarity of words and events, where we are actually interested in the probability of events to affect words. There are often multiple possible events that can act as explanations. Consider the following example. In 2010, several events related to Russia happened: New START (nuclear arms reduction treaty between the USA and Russia) was signed, there was a Winter Olympics, and the ROKS Cheonan (a South Korean warship) sunk. All relate to Russia, but only one appears to indicate a meaningful change to it—the New START event. Identifying the right events is a highly challenging task, as most usually all the candidate events are impactful events, related to the target word, and have an impact on various words due to their significance. As another example, Nelson Mandela's death is semantically the closest event to the word 'Clinton' during 2013, based on our projected embeddings. Though it is surely a powerful event and one that is related to many world leaders, it is hard to describe it as a turning point for either Bill or Hillary Clinton. Therefore, we attempt to *learn* which are the most relevant events for a given word. We present a classifier that receives an event and a word, and outputs the probability this event affected that word. This classifier functions as a predictor of the probability of an event e to cause a semantic change of a word w.

Training Data

To create training data for the classifier, we can use any embedding model that embeds both events and words, namely the global or the projected embeddings (Section 3 and 6.1). Given an event, we find terms affected by it and terms that are not.

Affected Terms: We limit the set of possible affected terms by an event and consider semantically similar terms to the event (we consider cosine similarity > 0.3). A term is considered to be affected by an event if the term's meaning changed during the time of the event, and did not change in the year before the event.

Unaffected Terms: Given an event, we sample terms from its semantically similar terms that were changed in the years *after* the event, and were not changed during the year it happened. This way, we try to capture terms that are related to the event but were changed due to other reasons.

Machine Learning Approach

We consider several supervised machine learning approaches, experimenting with random forest, SVM, neural networks, etc. We also devise several features leveraged by our classifiers:

- v_e and v_w, i.e., the embeddings of the event e and the word w. Any embedding model (Sections 3, 6.1) can be used.
- The semantic similarity between v_e and v_w.
- Categories of the event e, taken from DBpedia and represented using bag-of-words. Each event is associated with one or more categories in DBpedia (e.g., social event, sports, military conflict). The bag-of-words vector comprises the top 150 categories.
- Features that indicate the event's popularity: number of internal and external links in e's Wikipedia page, pageviews count of e's Wikipedia page[3]. Intuitively, a popular event might be impactful.

7 Experimental Setup

We briefly describe our dataset and embeddings before focusing on the evaluation.

7.1 Implementation Details

Embeddings: The global embeddings were created based on the Wikipedia dump of May 2016, using Word2Vec's skip-gram with negative sampling, with a window size of 10. Following Sherkat and Milios (2017), we perform a preprocessing step necessary for the embedding process to capture Wikipedia concepts and not just words: each inner link in the text (i.e., a link to a Wikipedia article) is replaced by an ID, so that every link to the same page is replaced by this page's ID. After filtering low-frequency words and concepts, we find 3.2M unique embeddings, of which 1.7M are concepts.

[3]We used Wikimedia Foundation's API to get pageviews of a single month (specifically October 2017).

For constructing temporal embeddings, we used the NYT archive[4], with articles from 1981 to 2016 (9GB of text in total). For each year of content, we created embeddings using Word2Vec's skip-gram with negative sampling, with a window size of 5 and a dimensionality of 140, using the Gensim library (Rehurek and Sojka, 2010). We filtered out words with less than 50 occurrences.

Events Data: We used DBpedia and Wikipedia as sources for events. First, we mined entities from DBpedia whose type is 'event' (i.e., `yago/Event100029378`, `Ontology/Event`), and that have an associated Wikipedia page and associated year of occurrence. Second, we mined 50K events from Wikipedia's monthly events pages[5] and retained the 30K events corresponding to our focus time period of 1981 to 2016. In this work, we focus on large, significant events, since these have the most potential to affect language (Chieu and Lee, 2004). Thus, we retained events with over 6000 monthly views (in October 2017) and over 15 external references. Our final dataset contains 1233 events, most related to armed conflicts, politics, disasters, and sports.

Classifier Dataset: To construct a "ground truth" dataset for our classifier, we find pairs of events (from the events dataset) and their affected and unaffected terms, as described in Section 6.3. For each event, we use either the global or projected embeddings to find 20 affected terms (there may be less, depending on the sensitivity of the change detection algorithm) and 20 unaffected terms. The dataset contains 21K pairs in total.

7.2 Experimental Methodology

We perform both qualitative and quantitative evaluations. First, we conduct user studies to capture the utility of our algorithms as they might be used in practice (e.g., in search engine results). Twenty evaluators participated using real events data (Section 7.1). We select a set of target words based on two criteria: popularity—so that most evaluators would know them and preferably parts of their history; and the number of significant related events—so that meaningful timelines would be produced. In practice, we selected 30 target words based on popularity (as expressed in the number of page views of the corresponding Wikipedia page).

[4] `http://spiderbites.nytimes.com/`

[5] For example, `https://en.wikipedia.org/wiki/Category:February_1992_events`

For a given target word, evaluators were presented with several timelines created by our algorithms and baseline methods (see below). While we evaluated both word and event descriptors for timelines, our evaluation is focused on events, as they proved to be more meaningful and interesting to the evaluators (see Section 8.2). Each timeline was accompanied by detailed descriptions and references. Our evaluators were asked to indicate whether an event was correct ('true') in its placement in the timeline and whether it was likely to have an impact on the target word. See Appendix A for a screenshot of the questionnaire used in the evaluation. Overall timeline quality was evaluated as described below.

Evaluation Metrics

Each timeline is evaluated using the following metrics (timelines with word descriptors are evaluated similarly—replacing 'event' with 'word'):

Accuracy: Fraction of events that are relevant to the target word (i.e., marked as true by the evaluators). $\frac{\#\text{true events}}{\#\text{true events}+\#\text{false events}}$

Relevance: How relevant the timeline is to the word. This is meant to approximate the precision metric. Relevance was indicated as a rank score on a scale from 1 to #timelines (#timelines being the number of timelines presented for the given word) and normalized as: $\frac{\text{relevance score}}{\#\text{timelines}}$

Missing Events: Evaluators were asked how many events they believed were missing from the timeline. We then normalize by the total number of events in the timeline. Intuitively, this measure is meant to approximate *1-recall*. $\frac{\#\text{missing events}}{\#\text{events in timeline}}$

Redundancy: Evaluators were asked how many events in the timeline are redundant (normalized by the total number of events in the timeline): $\frac{\#\text{redundant events}}{\#\text{events in timeline}}$

Ranking: Evaluators were asked to subjectively rank presented timelines from best to worst.

Effectiveness: Evaluators were asked to indicate their familiarity with the event history both before and after seeing the timeline. The difference between the two scores indicated the 'effectiveness' (as a soft measure of whether the timeline contained anything surprising or novel). The pre-evaluation score also served to measure the evaluator's familiarity with the topic.

Methods Compared

We perform experiments for the two building blocks of a timeline. First, we compare methods

of identifying turning points (Section 4). We approximate a gold standard for turning points by having evaluators mark years in which there is a turning point. This is determined by looking at all the events that took place during a specific year and approximating whether any of them is significant to the target word. We compare the identified years of our methods to this gold standard.

Second, we evaluated different ways to produce descriptors[6]:

WordDescriptors (Section 5): We use words as descriptors. For every year t, we select words that were added to the kNN of the target word during t (with $k = 20$, chosen empirically).

BaseEvents: A baseline method for event descriptors. We select events 'close' to the target word as descriptors—as measured by the frequency of the target word's appearance on the event's Wikipedia page.

WikiTimelines: Finally, we extract timelines from crowd-created Wikipedia timeline pages[7].

These baselines were compared to our algorithmic techniques:

ByWord and **ByKNN** are described in Section 6.2.

ByKNNGlobCls: We first use *ByKNN* to find 30 close events (determined empirically), and then use the events classifier (Section 6.3) to predict each event's influence. We rank the events by a combination of this prediction and the score of *ByKNN*. The classifier is trained on a dataset that was created using the global embeddings.

ByKNNCls: Similar to *ByKNNGlobCls*, with one difference: here the classifier's training set was created using the projected embeddings.

8 Results

8.1 Turning Point Evaluation

Comparing the two methods for turning point detection, we observed that 23% of the years detected by *Neighborhood* are false, compared to 15% by *EmbeddingSimilarity*. We believe this difference is due to the *Neighborhood* method capturing only the local neighborhood of the word, while an embedding can be more meaningful. For example, a word can change semantically not by altering its neighbors, but by moving in space towards other meanings, together with its neighbors.

8.2 Timeline Evaluation

We present the results of the timeline evaluation (Table 1), where the turning points are detected using the *EmbeddingSimilarity* method, as it reached the highest empirical performance (Section 8.1).

The *WordDescriptors* method achieves poor results, as expected. It is inaccurate and contains many redundant descriptors. For example, it generated the following descriptors for the word "Terror" during 2012-2014: Islamic terror, brutality, genocidal. They are all related to terror but do not help us deduce why the word 'Terror' was impacted, or what happened. Alternatively, the *ByWord* method performs much better. Looking at its generated timeline for 'Terror', we observe that it successfully identifies highly relevant events: the Benghazi attack (a terror attack against US government facilities in Libya), the mass shooting at Westgate Shopping Mall in Kenya, and the international military intervention against ISIL (the Islamic State organization). We find that *ByKNN* results and performance are similar to *ByWord*. As both methods consider the similarity between an event and a target word, we conjecture that the similarity function has a less impact on the timeline generation process. We empirically observe that in most cases a word close to an event would also be close to its neighbors.

The *WikiTimelines* method has the top accuracy. Given these timelines are manually created by domain experts, this is unsurprising. Nonetheless, the *ByKNNCls* method wins the three most important metrics: relevance, ranking, and effectiveness. Thus in the eyes of the evaluators, this method gives the most relevant timelines compared to all others, and maybe most importantly, provides novel information. As an example, the timeline created by *ByKNNCls* for 'Russia' contains several significant events that are missing from the same timeline created by *WikiTimelines*, such as Chernobyl disaster, the Revolutions of 1989 and the Dissolution of the Soviet Union.

The *ByKNNCls* method is more accurate than the other embedding-based methods, likely because it can filter out events that are identified by other methods but are in fact not impactful for the particular target word. However, it has a higher *Missing Events* score—suggesting true events are occasionally filtered out as well.

[6]Refer to https://github.com/guyrosin/generating_timelines for the source code.

[7]For example, https://en.wikipedia.org/wiki/Timeline_of_Russian_history

Method	Accuracy	Relevance	Missing	Redundancy	Ranking	Effectiveness
WordDescriptors	0.43	0.32	0.65	0.3	0.28	0.03
BaseEvents	0.49	0.76	0.04	**0.03**	0.72	0.23
WikiTimelines	**0.81**	0.67	**0.03**	**0.03**	0.65	0.07
ByWord	0.60	0.86	0.09	0.06	0.78	**0.33**
ByKNN	0.61	0.81	0.12	0.08	0.80	0.31
ByKNNGlobCls	0.63	0.62	0.38	0.1	0.53	0.17
ByKNNCls	0.67	**0.89**	0.20	0.09	**0.86**	**0.33**

Table 1: Timelines evaluation results. Methods and metrics described in Section 7.2.

To measure the correspondence between evaluators' answers, we calculated Kendall's Tau, which resulted in an average value of 0.6.

8.3 Events Classifier Evaluation

We experiment with several supervised approaches for the events classifier (Section 6.3) and evaluate their performance using stratified 10-fold cross-validation. In this evaluation, the projected embeddings were used to create the training and test sets and for creating the classifier's features, since this configuration was found to result in the best performance (Section 8.4). Specifically, the parameters (optimized using grid search) and the AUC are as follows:

Logistic regression produced an AUC of 0.75. **SVM** with RBF kernel and C=1.0 produced 0.97. **Random Forest** classifier with 800 trees produced 0.97 as well. **Neural Network** with a single hidden layer of 100 neurons and Adam as the optimization algorithm achieved the best performance, with an AUC score of 0.98.

8.4 Projection Contribution

We measure the embeddings projection's contribution (Section 6.1) to the tasks of timeline generation and learning influence of events on words.

Timeline Generation Performance

We refer the reader to Table 1 to discuss the comparison between *ByKNNCls* and *ByKNNGlobCls*. These methods are almost identical—both use our classifier for detecting significant events. They differ in how the classifier is trained. *ByKNNGlobCls*'s training set is created using global embeddings, while *ByKNNCls*'s training set is created using projected embeddings. We observe a significant difference in the performance of these two methods. *ByKNNCls* achieves the best performance of all methods. It has far fewer false negatives than *ByKNNGlobCls* and higher accuracy. The other important metrics—relevance, ranking, and effectiveness—show improved performance as well. We conclude that representing events using the projected embedding brings high performance boosts for this task.

Embeddings	Acc.	Rec.	Prec.	F1	AUC
Glob/Glob	0.63	0.66	0.62	0.64	0.69
Glob/Proj	0.74	0.73	0.74	0.74	0.81
Proj/Glob	0.76	0.76	0.75	0.76	0.86
Proj/Proj	**0.94**	**0.94**	**0.94**	**0.94**	**0.98**

Table 2: Classifier performance using global and projected embeddings for creating its features/dataset.

Events Classifier Performance

To better understand the embedding features on the events classifier's performance (Section 6.3) we compared the impact of global and projected embeddings. Additionally, the training data of the classifier can be generated using any embedding. Thus, we perform an empirical evaluation comparing all combinations of embeddings for representing the *features* and the *training data* employed by the classifier (Table 2). Using global embeddings, we find an AUC of 0.69. Using global embeddings for the features and projected embeddings for the dataset, or vice versa, resulted in AUC of around 0.84. Using the projected embeddings for both yielded an AUC of 0.98—significantly better than any other combination—with $p < 0.05$ (using a Wilcoxon signed-rank test).

8.5 Events Classifier Contribution

Our main goal in developing the classifier (Section 6.3) was to enable us to identify the relevant events that affect a given word. As presented in Table 1, the main drawback of *ByWord* and *ByKNN* is a high false positive ratio given

this task. The *ByKNNCls* method is more accurate than the other embedding-based methods, probably due to the classifier filtering out many events that are identified by the other methods but are in fact not impactful for the particular target word. For example, we observe several false events that appear in the 'Israel' timeline that was generated by the *ByKNN* method and are all ignored by *ByKNNCls*. Each one of them is related to Israel, but not significant enough to make an impact on it, e.g., the *anthrax letters* (2001) had "death to Israel" written inside them. *Hurricane Katrina* (2005) brought Israel to send a humanitarian aid delegation to New Orleans. Furthermore, the decrease in the false positive ratio results in high ratings of the *ByKNNCls* timelines. Looking at the results in Table 1, we observe significant differences in multiple metrics: relevance, ranking, and effectiveness (tested using paired t-test with $p < 0.05$).

8.6 Discussion

Three main factors seem to affect the performance of our approach. First, ambiguity harms performance. For creating timelines for ambiguous words, a contextual embedding approach would be necessary. We leave that for future work. For example, ambiguous target words such as 'Oil' had worse scores than other similar words (e.g., 'Tsunami' and 'Disaster') and than expected. Second, a sufficient amount of significant events relevant to the target word is crucial, otherwise, the timelines would be too short in the eyes of the evaluators or the end users. For example, 'ISIS' which is a relatively new organization, has a few significant relevant events and therefore had weak results in our evaluation. Third, the available amount of data about the target word makes a difference. The more the better, as the temporal embeddings would then be rich and meaningful. We observed worse performance for relatively rare words, such as 'Bombing', compared more common ones (e.g., 'Attack' and 'Russia').

9 Conclusions

In this work, we develop methods to model the evolution of language in relation to world events. We introduced the task of timeline generation, which is composed of two components: identifying turning points when semantic changes occur, and representing descriptors (i.e., words or events in our case). We presented several embeddings for the task and studied their effect. We find that our proposed method of projecting embeddings from a large, static model to a temporal one (i.e., from *Wikipedia* to the *New York Times*) yielded the best performance. Given several baselines we determined that a supervised approach leveraging the projected embeddings yields the best results. Using our method, high quality timeline generation can be done automatically and at scale.

Acknowledgements

We thank Prof. Eytan Adar for early feedback and comments.

References

Mario Alinei. 1995. Thirty-five definitions of etymology or: etymology revisited. In *On languages and Language-The Presidential Addresses of the 1991 Meeting of the Societas Linguae Europaea*, pages 1–26. Mouton de Gruyter, Berlin, New York.

Tim Althoff, Xin Luna Dong, Kevin Murphy, Safa Alai, Van Dang, and Wei Zhang. 2015. Timemachine: Timeline generation for knowledge-base entities. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 19–28. ACM.

Hosein Azarbonyad, Mostafa Dehghani, Kaspar Beelen, Alexandra Arkut, Maarten Marx, and Jaap Kamps. 2017. Words are malleable: Computing semantic shifts in political and media discourse. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1509–1518. ACM.

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 380–389. JMLR. org.

Pierpaolo Basile, Annalina Caputo, Roberta Luisi, and Giovanni Semeraro. 2016. Diachronic analysis of the italian language exploiting google ngram. *CLiC it*, page 56.

Hai Leong Chieu and Yoong Keok Lee. 2004. Query based event extraction along a timeline. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 425–432. ACM.

Dirk Geeraerts et al. 1997. *Diachronic prototype semantics: A contribution to historical lexicology*. Oxford University Press.

David Graus, Maria-Hendrike Peetz, Daan Odijk, Ork de Rooij, Maarten de Rijke, et al. 2013. yourhistory–semantic linking for a personalized

timeline of historic events. In *Workshop: LinkedUp Challenge at OKCon*.

William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1489–1501.

Adam Jatowt and Kevin Duh. 2014. A framework for analyzing semantic change of words across time. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 229–238. IEEE Press.

Tom Kenter, Melvin Wevers, Pim Huijnen, and Maarten De Rijke. 2015. Ad hoc monitoring of vocabulary shifts over time. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1191–1200. ACM.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*, pages 625–635. International World Wide Web Conferences Steering Committee.

Andrey Kutuzov, Lilja Øvrelid, Terrence Szymanski, and Erik Velldal. 2018. Diachronic word embeddings and semantic shifts: a survey. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1384–1397.

Jiwei Li and Claire Cardie. 2014. Timeline generation: Tracking individuals on twitter. In *Proceedings of the 23rd international conference on World wide web*, pages 643–652. ACM.

Rada Mihalcea and Vivi Nastase. 2012. Word epoch disambiguation: Finding how words change over time. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 259–263.

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1020–1029.

Matthias Orlikowski, Matthias Hartung, and Philipp Cimiano. 2018. Learning diachronic analogies to analyze concept change. In *Proceedings of the Second Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 1–11.

Octavian Popescu and Carlo Strapparava. 2013. Behind the times: Detecting epoch changes using large corpora. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 347–355.

Kira Radinsky, Krysta Svore, Susan Dumais, Jaime Teevan, Alex Bocharov, and Eric Horvitz. 2012. Modeling and predicting behavioral dynamics on the web. In *Proceedings of the 21st international conference on World Wide Web*, pages 599–608. ACM.

Radim Rehurek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer.

Guy D Rosin, Eytan Adar, and Kira Radinsky. 2017. Learning word relatedness over time. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1168–1178.

Maja Rudolph and David Blei. 2018. Dynamic embeddings for language evolution. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1003–1011. International World Wide Web Conferences Steering Committee.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2009. Semantic density analysis: Comparing word meaning across time and phonetic space. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 104–111. Association for Computational Linguistics.

Dafna Shahaf and Carlos Guestrin. 2010. Connecting the dots between news articles. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–632. ACM.

Ehsan Sherkat and Evangelos E Milios. 2017. Vector embedding of wikipedia concepts and entities. In *International Conference on Applications of Natural Language to Information Systems*, pages 418–428. Springer.

Terrence Szymanski. 2017. Temporal word analogies: Identifying lexical replacement with diachronic word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 448–453.

Tran Anh Tuan, Shady Elbassuoni, Nicoleta Preda, and Gerhard Weikum. 2011. Cate: context-aware timeline for entity illustration. In *Proceedings of the 20th international conference companion on World wide web*, pages 269–272. ACM.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*, pages 35–40. ACM.

Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 673–681. ACM.

Diversify Your Datasets: Analyzing Generalization via Controlled Variance in Adversarial Datasets

Ohad Rozen[1], **Vered Shwartz**[2,3], **Roee Aharoni**[1], **and Ido Dagan**[1]
[1]Computer Science Department, Bar-Ilan University, Ramat-Gan, Israel
[2]Allen Institute for Artificial Intelligence
[3]Paul G. Allen School of Computer Science & Engineering, University of Washington
{ohadrozen,roee.aharoni}@gmail.com, vereds@allenai.org, dagan@cs.biu.ac.il

Abstract

Phenomenon-specific "adversarial" datasets have been recently designed to perform targeted stress-tests for particular inference types. Recent work (Liu et al., 2019a) proposed that such datasets can be utilized for training NLI and other types of models, often allowing to learn the phenomenon in focus and improve on the challenge dataset, indicating a "blind spot" in the original training data. Yet, although a model can improve in such a training process, it might still be vulnerable to other challenge datasets targeting the same phenomenon but drawn from a different distribution, such as having a different syntactic complexity level. In this work, we extend this method to drive conclusions about a model's ability to learn and *generalize* a target phenomenon rather than to "learn" a dataset, by controlling additional aspects in the adversarial datasets. We demonstrate our approach on two inference phenomena – dative alternation and numerical reasoning, elaborating, and in some cases contradicting, the results of Liu et al.. Our methodology enables building better challenge datasets for creating more robust models, and may yield better model understanding and subsequent overarching improvements.

1 Introduction

To successfully recognize textual entailment (RTE; Dagan et al., 2013), also known as natural language inference (NLI) (MacCartney and Manning, 2008; Bowman et al., 2015), a system needs to model a broad range of inference phenomena. Pre-neural systems often included explicit components, such as engineered features or syntax-based transformations (e.g. Stern and Dagan, 2012; Stern et al., 2012; Bar-Haim et al., 2015), to address particular inference types such as syntactic, lexical, and logical inferences. Today's neural models do not explicitly model such inferences, but instead attempt to learn them implicitly from the training data. Despite their success on common NLI dataset, recent challenge datasets designed for probing different linguistic phenomena showed that neural models often fail on particular inference types, like recognizing semantic relations and negation (Poliak et al., 2018; Naik et al., 2018; Glockner et al., 2018).

Recently, Liu et al. (2019a) showed that when probing reveals a model's failure on a specific linguistic phenomenon, it is often possible to amend this failure. They suggested to fine-tune the model on (a training section of) the challenge dataset itself, in order to teach it to address the specific target phenomenon, or in other words, to "inoculate" it against the adversarial data. Inoculation has two possible outcomes. The first - a success to address the phenomenon after fine-tuning - suggests the original training set did not cover this phenomenon sufficiently ("blind spot" of the dataset). A failure, on the other hand, indicates an inherent model weakness to handle the target phenomenon. This was presented as a general methodology, and was demonstrated on the NLI task, among others.

The inoculation approach seems an appealing way to teach NLI models to properly address a broad range of inference phenomena, by training on a targeted inoculation dataset for each phenomenon. However, the methodology as suggested in Liu et al. (2019a) is not conclusive as to whether inoculation succeeded thanks to the model learning the target phenomenon in a *general* manner or due to overfitting the particular distribution of the inoculation data, possibly leveraging superficial cues or artifacts. Accordingly, successful inoculation may not reliably predict whether the model would successfully address the same target phenomenon when facing it on datasets drawn from different distributions.

In this paper, we extend the inoculation method-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 196–205
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

ology, analyzing the ability of models to generalize across different data distributions when addressing a specific inference phenomenon. In particular, we suggest varying both training and test distributions along several linguistic dimensions, like syntactic complexity or lexical diversity. In addition to indicating the model generalization ability, our methodology directs how to design the inoculation data in order to sufficiently cover the targeted phenomenon, when possible.

We demonstrate our methodology on two inference types, picked from the GLUE benchmark (Wang et al., 2019) diagnostic dataset: (1) dative alternation, a syntactic phenomenon, and (2) a specific type of numerical reasoning, pertaining to logical and arithmetic inference. To create our datasets, we introduce a templating method, by which we generated hundreds of synthetic examples from a single original sentence, while controlling the variance between the datasets.[1] We employ a recent NLI model, based on the pre-trained BERT masked language model (Devlin et al., 2019) fine-tuned on the MultiNLI dataset (Williams et al., 2018). For the dative alternation case, we find that the model struggles with generalizing over the syntactic dimension, requiring training over a relatively large variety of syntactically complex sentences. For the numerical reasoning case, we find that the model notably fails to generalize across diverse number ranges, a conclusion that might have been missed if we were to use only the original inoculation methodology. We hope our methodology will be adopted for additional NLP tasks, and specifically to a broader range of entailment inference types, as an avenue for developing robust NLI systems that can address specific inference phenomena.

2 Background

Neural NLI Models. Natural language inference is the task of identifying, given two text fragments, whether the second (*hypothesis*) can be inferred from the first (*premise*). While earlier models for these tasks relied on domain knowledge and lexical resources like WordNet (e.g MacCartney et al., 2008; Heilman and Smith, 2010), the release of the large-scale Stanford natural language inference dataset (SNLI; Bowman et al., 2015) shifted the focus to neural models which thrive given such large datasets. Typically, these models encode each of the premise and the hypothesis, combine them into a feature vector, and feed it into a classifier to make the entailment prediction. The encoding of the two sentences can be either independent of each other or dependent using an attention mechanism. These models typically do not rely on any external knowledge other than pre-trained word embeddings.

Contextualized Word Embeddings. Recently, the word embedding paradigm shifted from static token-based embeddings to dynamic context-sensitive ones. Notable contextual representations are ELMo (Peters et al., 2018b), BERT (Devlin et al., 2019), GPT (Radford et al., 2018) and XL-Net (Yang et al., 2019), which are pre-trained as language models on large corpora. Contextualized word embeddings have been used across a broad range of NLP tasks, outperforming the previous state-of-the-art models. Specifically, several works showed that they capture various types of linguistic knowledge, from syntactic to semantic and discourse relations (e.g. Peters et al., 2018a; Tenney et al., 2019; Shwartz and Dagan, 2019). Among many other tasks, NLI has also benefited from the use of contextualized word embeddings. The current state-of-the-art models use pre-trained contextualized word embeddings as their underlying representations, while fine-tuning on the NLI task (Liu et al., 2019b; Devlin et al., 2019). Despite their remarkable success on several datasets, it still remains unclear how these models represent the various linguistic phenomena required for solving the NLI tasks.

Existing Drawbacks and Challenge Datasets. Training an NLI model in this end-to-end manner assumes that any inference type involved in the sentence-level decision may be learned from the training data. However, recent work created challenge datasets which show that these models—when trained on the original NLI datasets—fail when they need to make inferences pertaining to certain linguistic phenomena, often ones which are not sufficiently represented in the training data. In these challenge datasets, a model is trained on the general NLI datasets, i.e. SNLI or the Multi-Genre Natural Language Inference datasets (MultiNLI; Williams et al., 2018). It is then used as a black box to evaluate on a given test set.

[1] All datasets and resources are available at https://github.com/ohadrozen/generalization.

Glockner et al. (2018) showed that substituting a single premise term with its hypernym (to create entailment examples) or a mutually exclusive term (for contradiction examples) challenges several pre-trained NLI models that performed well on the datasets on which they were trained. Naik et al. (2018) constructed a suite of "stress-tests", each pertaining to some linguistic phenomenon, and showed that NLI models fail on many of them (e.g. numerical reasoning, logical negations, etc.). Another line of work showed that NLI models may reach a surprising performance level on the NLI test sets just by exploiting artifacts in the generation of the hypotheses, rather than learning to model the complex entailment relationship (Gururangan et al., 2018; Tsuchiya, 2018; Poliak et al., 2018).

Fine-tuning on Challenge Datasets. Recently, Liu et al. (2019a) suggested that a model's failure to address a specific linguistic phenomenon may be attributed to one of the following cases: either the NLI training data does not sufficiently represent this phenomenon ("*dataset blind spot*") or the model is inherently incapable of learning to address this phenomenon. They suggested to fine-tune the NLI model on the specific challenge dataset in order to find out which case is currently observed. Specifically, in the case of a data blind spot, the performance on the challenge dataset is expected to improve after fine-tuning (i.e., the model is "inoculated" against the adversarial data). Otherwise, if the performance does not improve despite exposure to the phenomenon by fine-tuning, this may be an inherent weakness of the model. Finally, an additional possible outcome is that the performance on the original NLI test set is severely hurt after fine-tuning on the specific phenomenon, which may be due to overfitting.

3 Methodology

We extend the inoculation approach of Liu et al. (2019a) by additionally controlling for finer-grained dimensions of the training and test data. For a given inference type (Section 3.1), our methodology consists of the following steps. (1) First, we extract *premises* in the MultiNLI (Williams et al., 2018) training set that include the targeted linguistic phenomenon (Section 3.2). (2) For each found premise, we generate multiple diverse variations using our templating method (Section 3.3). (3) After generating diverse premises, we generate multiple matching *synthetic hypotheses* using a templating method (Section 3.4). As we generate synthetic hypotheses, we can make sure the premise-hypothesis pairs differ along our proposed diversity dimensions. (4) Finally, we define the train and test sets so that the variance between them is controlled with respect to the different dimensions (Section 3.5). This facilitates probing the success of the model to generalize a given inference type with respect to a specific dimension of the data.

3.1 Inference Types

We focus on the following two inference types from the diagnostic set of the GLUE benchmark (Wang et al., 2019) as test cases for our methodology.

Dative Alternation. This inference type refers to the alternation between a double-object construction ("*I baked him a cake*") and a prepositional indirect-object construction ("*I baked a cake for him*").

Numerical Reasoning. We focus on sentences relating to numbers by the relational phrases "*more than*" and "*less than*", normalizing all numbers to numerals. For example, "*There are **3** apples on the table*" entails "*There are **more than 2** apples on the table*"

3.2 Premise Extraction

For a given inference type, we start by finding premises in the MultiNLI train set that include the targeted linguistic phenomenon. We then construct templates based on these premises which are later used to generate synthetic premises and hypotheses. We do so in a semi-automatic way: we first use simple heuristics to track good candidates, and then manually select those that can be used as premises of NLI pairs that include the linguistic phenomenon in focus. For example, to track premises for the numerical reasoning inference type, we search for premises containing numbers and then choose the ones in which adding *more than* or *less than* before the number would keep the premise grammatical and coherent (e.g. "*the U.S. economy added **45** million jobs.*"), or ones which already include these terms before the number (e.g. "*The Citigroup deal, from beginning to end, took **less than 5** weeks.*"). We also make sure that we have enough diversity in the premise

	Dative Alternation	
(1)	**Extracted Premise:**	*[Even our noble Saudi allies] [aren't willing to] lend [us] [their air bases].*
(2)	**Premise Template:**	*ARG_1 ARG_2 lend **ARG_3** ARG_4.*
(3)	**Hypothesis Template (Ent. #1):**	*ARG_1 ARG_2 lend ARG_4 **to** **ARG_3**.*
(4)	**Gen. Premise:**	*[The allies across the sea] [have promised to] lend [**Italy**] [some of their land].*
(5)	**Gen. Hypothesis (Ent. #1):**	*The allies across the sea have promised to lend some of their land **to Italy**.*
(6)	**Gen. Hypothesis (Ent. #2):**	*The allies across the sea have promised to lend some of their land.*
(7)	**Gen. Hypothesis (Cont.):**	*The allies across the sea have promised to lend **Italy**.*
	Numerical Reasoning	
(8)	**Extracted Premise:**	*[The Citigroup deal], [from beginning to end], [took] **less than** 5 [weeks].*
(9)	**Premise Template:**	*ARG_1, ARG_2, ARG_3 **REL_p** NUM_p ARG_4.*
(10)	**Hypothesis Template (Ent.):**	*ARG_1, ARG_2, ARG_3 **more than** $NUM_{smaller}$ ARG_4.*
(11)	**Gen. Premise:**	*[My marriage], [despite much frustration], [lasted] **more than** 7 [years].*
(12)	**Gen. Hypothesis (Ent.):**	*[My marriage], [despite much frustration], [lasted] **more than** 2 [years].*
(13)	**Gen. Hypothesis (Cont.):**	*[My marriage], [despite much frustration], [lasted] **less than** 5 [years].*
(14)	**Gen. Hypothesis (Neutral):**	*[My marriage], [despite much frustration], [lasted] 8 [years].*

Table 1: Examples for the premise-hypothesis generation process (notations are explained in Sections 3.3 and 3.4): (a) Premises are extracted from the MultiNLI train set (rows 1 and 8) (b) Premise templates are manually created (rows 2 and 9) (c) Hypothesis templates are automatically generated using the premise templates (rows 3 and 10) (d) New premises are automatically generated by instantiating them with the turkers' answers (rows 4 and 11) (e) new hypotheses with same instantiations are generated (rows 5-7 and 12-14).

length and syntactic complexity (see Section 3.5). Rows 1 and 8 in Table 1 exemplify such premises for the dative alternation and numerical reasoning inference types.

3.3 Premise Generation

To isolate the lexical dimension from the syntactic one, for each target phenomenon we synthesize multiple new premises, all sharing a similar syntactic structure by construction. To do so, we manually generate a *premise template* by replacing at least four spans in the premise with arguments ARG_i as placeholders. We do so while keeping the words related to the phenomenon in focus within the template. For example, from the premise "*Even our noble Saudi allies aren't willing to lend us their air bases.*", we generated the template "ARG_1 ARG_2 *lend* ARG_3 ARG_4." (see rows 2 and 9 in Table 1). We then let crowdsourcing workers instantiate each of the arguments to create new sentences. For the instantiations to later construct coherent sentences with high likelihood, we ask the workers to instantiate each argument separately, leaving the rest of the sentence unchanged, in a way that yields a new grammatical and coherent sentence that can make sense in some possible made up context (e.g. "*Even our noble Saudi allies ⟨* ***span to fill in*** *⟩ lend us their air bases.*"). To maintain similar sentence lengths and structures, we limit the instantiations to be at most one word longer or shorter than the original spans. For each argument we collected 6 instantiations which were manually validated for grammaticality and semantic coherence. We used all possible combinations of instantiations to generate hundreds of premises per template (rows 4 and 11 in Table 1). The annotation task was performed in Amazon Mechanical Turk, where to control the quality of the workers, we required that they have at least 98% acceptance rate for prior HITs. We paid $2.5 for two instantiations of all arguments in a given sentence (at most 14 instantiations per sentence).

3.4 Hypothesis Generation

For each *premise template* we automatically generate multiple *hypothesis templates* for each entailment label (entailment, neutral and contradiction), which differ from the premise only in the phenomenon in focus, as detailed for each inference type below.

Dative Alternation. We generate the entailed hypothesis templates by applying the inference type. Specifically, for entailing dative alternation, we either switch to the alternate constructions (row 5 in Table 1) or remove the first argument after the dative verb (row 6). For contradicting hypotheses templates, we remove the second argument, creating a grammatical yet contradictory hypothesis template (row 7). For each premise template we therefore generate 2 entailment hypotheses and 1 contradictory (no neutral).

Numerical Reasoning. For a premise consisting of a target numeric value NUM_p preceded by a relational expression REL_p $\in$

$\{less\ than, more\ than, \emptyset\}$, we generate multiple hypotheses by replacing the target number by a random number (from a given target range, see further below) and the relational expression by each of the expressions. The sentence-pair label is determined by the relation between the numeric expressions in the premise and the hypothesis, and may be any of entailment, neutral, or contradiction. For example, consider the premise template "[*The union*] [*has*] more than 4 [*thousand members*] [*in Canada*]". Replacing the numeric expression "*more than 4*" by "3" yields contradiction, while the substitute "*more than 3*" is entailing, and "*more than 5*" is neutral. This way, for each premise template we generate 22 different hypotheses: 4 entailment, 6 neutral and 12 contradiction. We used numeric values within the range 2-999.

3.5 Controlled Data Splits

The main motivation for our data splits is to control the variance between the train and test sets with respect to certain data dimensions. Specifically, for both inference types, we create a variance along the syntactic complexity dimension. Based on the premise template, we divided each dataset into 3 subsets with different syntactic complexity levels: *simple*, *medium* and *complex*, denoted by S, M and C respectively. We do so according to two criteria: sentence length and the depth in the constituency parsing tree in which the inference type occurs, using the Stanford Parser (Manning et al., 2014) (see Table 2).[2]

We also create a variance along different lexical dimensions. From the *simple* subset S we generate two additional subsets S_{Lex1} and S_{Lex2} in the following way. For each *simple* template, we first split its original instantiations into two groups, and then instantiate each such group separately into the template, creating two sets of instantiations of the same template $s_1 \in S_{Lex1}$ and $s_2 \in S_{Lex2}$, which are syntactically similar by construction, yet lexically different. We repeat this process for the *complex* subset as well to create C_{Lex1} and C_{Lex2}. We further split the dative alternation datasets lexically by the main verb, which allows testing how well the model generalizes this inference type for different dative verbs. This is done by changing the main dative verb in S_{Lex2} and C_{Lex2} creating new subsets S'_{Lex2} and C'_{Lex2}.[3] We split the numerical reasoning dataset similarly using different numerical ranges in the training and test sets.

In Section 4 we experiment with various splits of the training and test sets, which allow us to test the model's generalization over the various dimensions. For example, we test whether the model can learn an inference type on syntactically *simple* examples and generalize it to *complex* ones. See Table 2 for the statistics of each dataset.

	Simple	Medium	Complex	All
Number of Premise Templates				
Datives	10	9	9	28
Numbers	9	12	9	30
Number of Examples				
Datives	21K	36K	34K	91K
Numbers	181K	239K	182K	602K

Table 2: Statistics of the dative alternation and numerical reasoning datasets divided by syntactic complexity level.

4 Experiments

We use a standard model based on BERT (Devlin et al., 2019) as our NLI model. Specifically, we used the base-uncased pre-trained model from the pytorch-pretrained-bert library[4], and fine-tuned it for the NLI task on MultiNLI.[5] We conduct several experiments. First, we use our datasets for a typical probing task, i.e. testing how well the model performs on each inference type, *without being trained to address the specific inference type* (Section 4.1). Then, similarly to Liu et al. (2019a), we test the model's ability to learn each inference type by *further fine-tuning on specific examples for it* (Section 4.2). Finally, incorporating our innovation, we analyze the model's generalization ability by introducing variance in the proposed data dimensions between the train and test sets (Section 4.3).

4.1 Probing

We randomly selected 4,000 examples from each of the *simple*, *medium* and *complex* datasets, with

[2] For dative alternation we consider the depth of the dative verb, while for the numeric reasoning data we look at the depth of the number. We consider premise templates with less than 16 words and depth < 4 as *simple*, more than 25 words and depth > 6 as *complex*, and the rest as *medium*.

[3] For each template we choose a new dative verb that keeps the sentence coherent and grammatical.

[4] https://github.com/huggingface/pytorch-pretrained-bert

[5] The MultiNLI dataset has domain-matched and mismatched development data. We use "matched" for our testing.

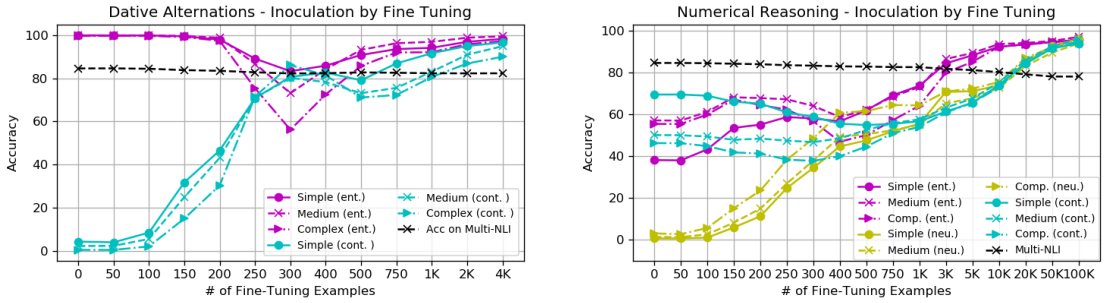


Figure 1: Average test accuracy on the dative alternation dataset (left) and the numerical reasoning dataset (right) as a function of number of training examples, divided by label. The black lines represent the average accuracy of the model on MultiNLI matched development set after fine-tuning. For numerical reasoning we use a larger range on the x-axis to capture the near-perfect performance for 100k examples.

Complexity	Ent.	Neutral	Cont.	All
MultiNLI Matched Dev Set				
All	83.56	84.12	86.37	**84.66**
Dative Alternation				
Simple	100	-	4.22	**52.63**
Medium	100	-	2.16	**49.27**
Complex	99.77	-	0.36	**50.45**
All	99.92	-	2.25	**50.78**
Numerical Reasoning				
Simple	38.14	0.66	69.53	**45.04**
Medium	57.14	1.36	50.14	**38.11**
Complex	55.48	3.04	46.26	**36.15**
All	50.25	1.69	55.31	**39.77**

Table 3: Accuracy of the model trained only on MultiNLI on our datasets, which are used as probing datasets. The Complexity column refers to the syntactic complexity of the sentences.

balanced labels, to serve as a test set. Table 3 shows the accuracy of the model on each test set.

On our dative alternation dataset, the accuracy on our test sets is substantially lower than on the MultiNLI development set (50.78% versus 84.66% respectively), suggesting that the model has not learned to address this inference type from the MultiNLI training data. The model has very high accuracy on the entailment examples, while close to zero on the contradiction ones. This is understandable considering that the sentence-pairs in this dataset by construction have high lexical overlap between the premise and hypothesis, leading the model to default to almost always predicting entailment.

On the numerical reasoning dataset, the model also seems to fail on this inference type with test set accuracy much lower than on the MultiNLI development set, suggesting that the model hasn't learned to address this inference type as well. The model has relatively low accuracy on the entailment and contradiction examples while close to zero accuracy on the neutral ones. This is due to the fact that the model classifies sentence-pairs with high lexical overlap but with a different numerical phrase as either entailment or contradiction, but almost never as neutral.

4.2 Fine Tuning

We follow Liu et al. (2019a) and fine-tune the model on the phenomenon-specific examples, testing how many training examples the model needs to observe before it performs reasonably well on this inference type.[6]

We split each dataset to training (77%) and test (23%) sets such that the same template is not used in both training and test. Each set consists of templates from all syntactic complexities, and a balanced number of examples from each label. We experiment with a different number of training examples ranging from 0 to 4,000 for the dative alternation sets and from 0 to 100,000 for numerical reasoning. We repeat this experiment five times with different training and test splits, while also testing the performance on the original MultiNLI matched development dataset. We report in Figure 1 the average accuracy across runs as a function of the number of training examples. In both datasets, fine-tuning greatly improves the performance.

On the dative alternation data, fine-tuning brings the performance on contradiction from 0 to 90%, suggesting the model can now distinguish well between the entailing and contradictory examples, reaching similar accuracy on both. The

[6]To fine-tune BERT, we use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of $7 \cdot 10^{-7}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and L_2 weight decay of 0.01.

good performance on this inference type indicates a blind spot in the MultiNLI dataset rather than a model weakness. As expected, the model reaches slightly better performance on examples with simpler syntactic structure than those with complex ones. Fine-tuning with 4,000 examples reduces the performance on the MultiNLI development set in about 2%. As Liu et al. suggested, this could result from the distribution of our dataset deviating from the distribution of the MultiNLI dataset, and possibly from having the original model overfitting to that distribution.

With respect to numerical reasoning, though only after a relatively large number of 10,000 training examples, the model seems to succeed in learning the phenomenon. According to Liu et al. (2019a), this result suggests that our challenge dataset did not reveal a weakness in the model, but instead a blind spot in the original dataset. Yet, this conclusion is challenged in the next subsection. In the numerical reasoning case we notice an even larger decrease in the performance on MultiNLI after fine-tuning - up to 6.5%. This may again result from different distributions across the datasets, influencing the performance more intensely due to the larger number of training examples.

4.3 Generalization

We now analyze the model's ability to generalize for each inference type across various data dimensions, using our proposed methodology. As we will see, this type of analysis yields additional, more elaborate and sometimes contradictory insights, which are not attainable by the prior methodologies that we applied in the previous two subsections.

Dative Alternation. First, we test the model's generalization ability at the syntactic complexity dimension, by training it on a dataset belonging to one category of syntactic complexity (among *simple*, *medium* and *complex*; see Section 3.5), and testing it on a dataset belonging to either *simple* or *complex*. We make sure examples in the training and test sets were generated by different templates.

The left side of Figure 2 displays the performance on the various experiments, revealing an interesting pattern: on the *simple* syntax test set, good performance is attainable regardless of the training set, while for the *complex* syntax test set, training on simple syntax performs inferiorly to training on complex syntax. This suggests that although the model is able to learn the dative alternation phenomenon and generalize to a certain extent, the model does not learn the phenomenon on its own, decoupled from learning argument positions, but rather it needs to be trained with dative alternation examples of high syntactic complexity to perform well.

The second data dimension we test is lexical diversity. We test the model's ability to generalize across syntactically-similar examples with a different main dative verb. To isolate the lexical aspect from other aspects, we fix the syntax by using the same templates for training and testing. For the *simple* category, we train the model on the S_{Lex1} subset and test it on both the S_{Lex2} subset with the same main dative verb, and on S'_{Lex2} with a different main verb that has not been seen in the training examples (see Section 3.5). We repeat the same process for the *complex* category.[7]

The right side of Figure 2 shows that when tested on the same syntactic complexity level as seen during training, the performance remains similar regardless of the similarity between the train and test dative verbs. This suggests that the model generalizes well on the lexical dimension and learns to recognize the dative alternation inference independently of the specific verb. We also observe that the model generalizes more easily from examples with simple syntax: on this category, the performance gap between the two test sets S_{Lex2} and S'_{Lex2} is smaller than the gap between the graphs of the complex category. This suggests the conclusion that unlike syntactic diversity, a large lexical diversity is not necessary when inoculating for dative alternation.

Numerical Reasoning. Again, we test the model's generalization ability with respect to syntactic complexity by splitting the train and test sets based on this dimension (left side of Figure 3). As opposed to dative alternation, here the gap between the performance when testing examples with more complex syntax than the training set and the performance when testing on simpler examples is rather small after enough training examples (up to 3.2% difference after 3,000 examples). We conjecture that the model learns to identify local patterns (e.g. "*more than* X") while the rela-

[7] For both S_{Lex1} and C_{Lex1} we sample 256 examples from each of 5 manually chosen templates from the related category.

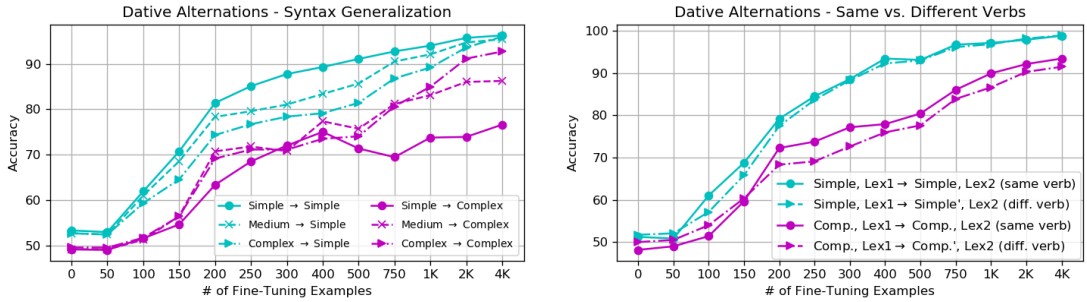


Figure 2: Test accuracy on the dative alternation dataset over the syntactic dimension (left) and average performance on the lexical dimension (right) as a function of number of training examples. The larger gaps on the left graph in comparison to the much smaller gaps on the right graph indicate a limited generalization ability over the syntactic dimension and a better one over different dative verbs

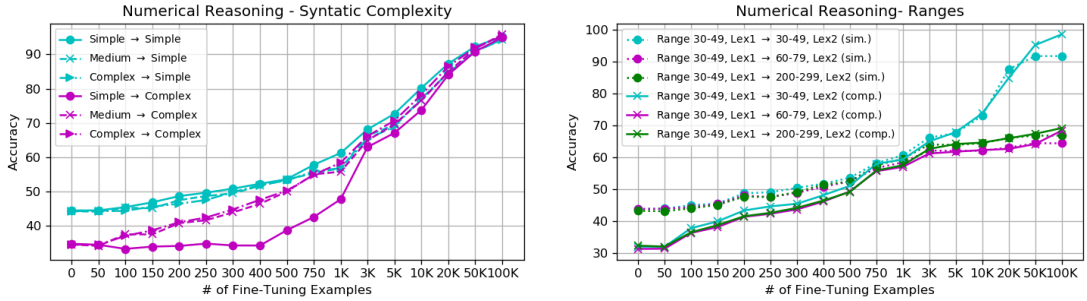


Figure 3: Test accuracy on the numerical reasoning dataset over the syntactic dimension (left) and on the range dimension (right) as a function of number of training examples. The gaps on the right graph comparing to the convergence on the left one suggest a good generalization ability over the syntactic dimension and a poor one over different number ranges.

tionships between them are less dependent on the global syntactic structure of the sentence. Moreover, the difference between the premise and the hypothesis is lexical and local (i.e. replacing the relational operator and the number), while their syntax remains otherwise identical, shifting the model's focus away from the syntax.

Regarding lexical diversity, we test whether the model can be trained on one range of numbers and perform well when tested on another. To that end, we populate the number placeholders in the templates with randomly sampled numbers from within a certain range, among 30-49, 60-79 and 200-299. We train on the first range and test on each of the ranges. To isolate the numeric aspect from the syntactic one, we fix the syntax by using the same templates for training and testing, while using different argument instantiations (as we did for the dative alternation when testing for lexical diversity of the dative verb).

The right side of Figure 3 shows the performance of each model as a function of training examples. Training and testing on the same range yields substantially better performance, while testing on a different range reaches accuracy of less than 70% even after 20,000 training examples, and seems to reach saturation. We also repeated the same experiment with number range of 1000-9999 in the test sets, resulting in a graph very similar to the 200-299 range. This indicates an inherent weakness of the model to learn the phenomenon and generalize it over different number ranges. Given that the number of training examples is limited, it might be challenging to inoculate the model to perform well on a wide variety of challenge datasets for this phenomenon. The success within the same (narrow) number range, when training gets large enough, might suggest that the model mostly memorizes specific number pairs and the arithmetic relation between them, rather than learning the arithmetic rules. These insights contradict the conclusion of the original inoculation analysis of a blind spot in the original dataset (in Section 4.2).

5 Conclusions

We presented a methodology to analyze the ability of NLI models to learn a specific inference

phenomenon and successfully generalize its modeling to datasets drawn from different distributions. By controlling the differences between the training and test sets along syntactic and lexical data dimensions, we were able to analyze how well the model generalizes with respect to each phenomenon over the different dimensions. We demonstrated our methodology on a standard model based on BERT, focusing on dative alternation and numerical reasoning. We found that high syntactic complexity is necessary for teaching dative alternations, while being less important for numerical reasoning. We also showed that the model is incapable of generalizing over different number ranges for numerical reasoning, indicating an inherent modeling weakness.

We suggest that our work opens promising as well as challenging research directions. A natural direction for future work would be to apply our methodology to a broader range of inference types and data dimensions. This would enable extensive analysis of NLI models' learning and generalization abilities, and may yield models that can truly address a range of inference phenomena in a fairly general manner. One question that still remains open at this point regards the models' ability to handle *multiple* inference phenomena within the same example, which is more representative of real-world scenarios.

Finally, we observed that fine-tuning the model on the phenomenon-specific data sometimes yields a decrease in the performance on the original dataset. One potential direction to avoid this in the future would be to perform multi-task learning rather than fine-tuning on the phenomenon-specific data. Better scheduling of multi-task training as presented by Kiperwasser and Ballesteros (2018) may also reduce the performance loss in such scenarios.

Acknowledgments

We would like to thank Ori Shapira for assisting in data analysis, and the anonymous reviewers for their constructive comments. This work was supported in part by the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1), by a grant from Reverso and Theo Hoffenberg, and by the Israel Science Foundation (grant 1951/17).

References

Roy Bar-Haim, Ido Dagan, and Jonathan Berant. 2015. Knowledge-based textual inference via parse-tree transformations. *J. Artif. Int. Res.*, 54(1):1–57.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota. Association for Computational Linguistics.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. In *The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, New Orleans, Louisiana.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1011–1019, Stroudsburg, PA, USA. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

Nelson F Liu, Roy Schwartz, and Noah A Smith. 2019a. Inoculation by fine-tuning: A method for analyzing challenge datasets. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019b. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504.

Bill MacCartney, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 802–811, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 521–528, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018b. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. 2018. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 67–81, Brussels, Belgium. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*.

Vered Shwartz and Ido Dagan. 2019. Still a pain in the neck: Evaluating text representations on lexical composition. In *Transactions of the Association for Computational Linguistics (TACL)*, page (to appear).

Asher Stern and Ido Dagan. 2012. Biutee: A modular open-source system for recognizing textual entailment. In *Proceedings of the ACL 2012 System Demonstrations*, ACL '12, pages 73–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

Asher Stern, Roni Stern, Ido Dagan, and Ariel Felner. 2012. Efficient search for transformation-based inference. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 283–291, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you Learn from Context? Probing for Sentence Structure in Contextualized Word Representations. In *International Conference on Learning Representations*.

Masatoshi Tsuchiya. 2018. Performance impact caused by hidden bias of training data for recognizing textual entailment. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Fully unsupervised crosslingual semantic textual similarity metric based on BERT for identifying parallel data

Chi-kiu Lo and **Michel Simard**
NRC-CNRC
National Research Council Canada
1200 Montreal Road, Ottawa, Ontario K1A 0R6, Canada
{Chikiu.Lo|Michel.Simard}@nrc-cnrc.gc.ca

Abstract

We present a fully unsupervised crosslingual semantic textual similarity (STS) metric, based on contextual embeddings extracted from BERT – Bidirectional Encoder Representations from Transformers (Devlin et al., 2019). The goal of *crosslingual STS* is to measure to what degree two segments of text in different languages express the same meaning. Not only is it a key task in crosslingual natural language understanding (XLU), it is also particularly useful for identifying parallel resources for training and evaluating downstream multilingual natural language processing (NLP) applications, such as machine translation. Most previous crosslingual STS methods relied heavily on existing parallel resources, thus leading to a circular dependency problem. With the advent of massively multilingual context representation models such as BERT, which are trained on the concatenation of non-parallel data from each language, we show that the deadlock around parallel resources can be broken. We perform intrinsic evaluations on crosslingual STS data sets and extrinsic evaluations on parallel corpus filtering and human translation equivalence assessment tasks. Our results show that the unsupervised crosslingual STS metric using BERT without fine-tuning achieves performance on par with supervised or weakly supervised approaches.

1 Introduction

Crosslingual semantic textual similarity (STS) (Agirre et al., 2016a; Cer et al., 2017) aims at measuring the degree of meaning overlap between two texts written in different languages. It is a key task in crosslingual natural language understanding (XLU), with applications in crosslingual information retrieval (Franco-Salvador et al., 2014; Vulić and Moens, 2015), crosslingual plagiarism detection (Franco-Salvador et al., 2016a,b), etc. It is also particularly useful for identifying parallel resources (Resnik and Smith, 2003; Aziz and Specia, 2011) for training and evaluating downstream multilingual NLP applications, such as machine translation systems.

Unlike in crosslingual textual entailment (Negri et al., 2013) or crosslingual natural language inference (XNLI) (Conneau et al., 2018), which are directional classification tasks, in crosslingual STS, continuous values are produced, to reflect a range of similarity that goes from complete semantic unrelatedness to complete semantic equivalence. Machine translation quality estimation (MTQE) (Specia et al., 2018) is perhaps the field of work that is the most related to crosslingual STS: in MTQE, one tries to estimate translation quality, by comparing an original source-language text with its machine translation. In contrast, in crosslingual STS, neither the direction nor the origin (human or machine) of the translation is taken into account. Furthermore, MTQE also typically considers the fluency and grammaticality of the target text; these aspects are usually not perceived as relevant for crosslingual STS.

Many previous crosslingual STS methods rely heavily on existing parallel resources to first build a machine translation (MT) system and translate one of the test sentences into the other language for applying monolingual STS methods (Brychcín and Svoboda, 2016). Methods that do not rely explicitly on MT, such as that in Lo et al. (2018), still require parallel resources to build bilingual word representations for evaluating crosslingual lexical semantic similarity. It is clear that there is a circular dependency problem on parallel resources.

Massively multilingual context representation models, such as MUSE (Conneau et al., 2017), BERT (Devlin et al., 2019), and XLM (Lample and Conneau, 2019), that are trained in an unsupervised manner with non-parallel data from each

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 206–215
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

language, have shown improved performance in XNLI classification tasks using task-specific fine-tuning.

In this paper, we propose a crosslingual STS metric based on fully unsupervised contextual embeddings extracted from BERT without fine-tuning. In an intrinsic crosslingual STS evaluation and extrinsic parallel corpus filtering and human translation error detection tasks, we show that our BERT-based metric achieves performance on par with similar metrics based on supervised or weakly supervised approaches. With the availability of the multilingual context representation models, we show that the deadlock around parallel resources for crosslingual textual similarity can be broken.

2 Crosslingual STS metric

Our crosslingual STS metric is based on YiSi (Lo, 2019). YiSi is a unified adequacy-oriented MT quality evaluation and estimation metric for languages with different levels of available resources. Lo et al. (2018) showed that YiSi-2, the crosslingual MT quality estimation metric, performed almost as well as the "MT + monolingual MT evaluation metric (YiSi-1)" pipeline for identifying parallel sentence pairs from a noisy web-crawled corpus in the *Parallel Corpus Filtering* task of WMT 2018 (Koehn et al., 2018b).

To measure semantic similarity between pairs of segments, YiSi-2 proceeds by finding alignments between the words of these segments that maximize semantic similarity at the lexical level. For evaluating crosslingual lexical semantic similarity, it relies on a crosslingual embedding model, using cosine similarity of the embeddings from the crosslingual lexical representation model. Following the approach of Corley and Mihalcea (2005), these lexical semantic similarities are weighed by *lexical specificity* using inverse document frequency (IDF) collected from each side of the tested corpus.

As an MTQE metric, YiSi-2 also takes into account fluency and grammatically of each side of the sentence pairs using bag-of-ngrams and the semantic parses of the tested sentence pairs. But since crosslingual STS focuses primarily on measuring the meaning similarity between the tested sentence pairs, here we set the size of ngrams to 1 and opt not to use semantic parses in YiSi-2. In addition, rather than compute IDF weights $w(e)$ and $w(f)$ for lexical units e and f in each language directly on the texts under consideration, we rely on precomputed weights from monolingual corpora $\mathbb{E}$ and $\mathbb{F}$ of the two tested languages.

The YiSi metrics are formulated as an F-score: by viewing the source text as a "query" and the target as an "answer", precision and recall can be computed. Depending on the intended application, precision and recall can be weighed differently. For example, in MT evaluation applications, we typically assign more weight to recall ("every word in the source should find an equivalent in the target"). For this application, we give equal weights to precision and recall.

Thus, the crosslingual STS of sentences **e** and **f** using YiSi-2 in this work can be expressed as follows:

$$
\begin{aligned}
v(u) &= \text{embedding of unit } u \\
s(e,f) &= cos(v(e), v(f)) \\
w(e) &= idf(e) = log(1 + \frac{|\mathbb{E}| + 1}{|\mathbb{E}_{\exists e}| + 1}) \\
w(f) &= idf(f) = log(1 + \frac{|\mathbb{F}| + 1}{|\mathbb{F}_{\exists f}| + 1}) \\
\text{precision} &= \frac{\sum_{e \in \mathbf{e}} \max_{f \in \mathbf{f}} w(e) \cdot s(e,f)}{\sum_{e \in \mathbf{e}} w(e)} \\
\text{recall} &= \frac{\sum_{f \in \mathbf{f}} \max_{e \in \mathbf{e}} w(f) \cdot s(e,f)}{\sum_{f \in \mathbf{f}} w(f)} \\
\text{YiSi-2} &= \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}
\end{aligned}
$$

where $s(e,f)$ is the cosine similarity of the vector representations $v(e)$ and $v(f)$ in the bilingual embeddings model.

In the following, we present the approaches we experimented with to obtain the crosslingual embedding space in supervised, weakly supervised and unsupervised manners.

2.1 Supervised crosslingual word embeddings with *BiSkip*

Luong et al. (2015) proposed *BiSkip* (with open source implementation `bivec`[1]) to jointly learn bilingual representations from the context cooccurrence information in the monolingual data and the meaning equivalent signals in the parallel data. It trains bilingual word embeddings with the objective to preserve the clustering structures of

[1] https://github.com/lmthang/bivec

words in each language. We train our crosslingual word embeddings using `bivec` on the parallel resources as described in each experiment.

2.2 Weakly supervised crosslingual word embeddings with *vecmap*

Artetxe et al. (2016) generalized a framework to learn the linear transformation between two monolingual word embedding spaces by minimizing the distances between equivalences listed in a collection of bilingual lexicons (with open source implementation `vecmap`[2]). We train our monolingual word embeddings using `word2vec`[3] (Mikolov et al., 2013) on the monolingual resources and then learn the linear transformation of the two monolingual embedding space using `vecmap` on the dictionary entries as described in each experiment.

2.3 Unsupervised crosslingual contextual embeddings with multilingual BERT

The above two mentioned embedding models produce static word embeddings that captures the semantic space to represent the training data. The shortcoming of these static embedding models is that they provide the same embedding representation for the same word without reflecting the context variation of them being used in different sentences. In contrast, BERT (Devlin et al., 2019) uses a bidirectional transformer encoder (Vaswani et al., 2017) to capture the sentence context in the output embeddings, such that the embedding for the same word unit in different sentences would be different and better represented in the embedding space. Multilingual BERT model is trained on the Wikipedia pages of 104 languages with a shared subword vocabulary. Pires et al. (2019) showed multilingual BERT works well on different monolingual NLP tasks across different languages.

Following the recommendation in Devlin et al. (2019), we use embeddings extracted from the ninth layer of the pretrained multilingual cased BERT-Base model[4] to represent subword units in the two sentences in assessment for the crosslingual lexical semantic similarity.

[2] https://github.com/artetxem/vecmap

[3] https://code.google.com/archive/p/word2vec/

[4] https://github.com/google-research/bert

3 Experiment on crosslingual STS

We first evaluate the performance of YiSi-2 on the intrinsic crosslingual STS task, before testing its ability on the downstream task of identifying parallel data.

3.1 Setup

We use data from the SemEval-2016 Semantic Textual Similarity (STS) evaluation's crosslingual track (task1) (Agirre et al., 2016b), in which the goal was to estimate the degree of equivalence between pairs of Spanish-English bilingual fragments of text.[5] The test data is partitioned into two evaluation sets: the *News* data set has 301 pairs, manually harvested from comparable Spanish and English news sources; the *Multi-source* data set consists of 294 pairs, sampled from English pairs of snippets used in the SemEval-2016 monolingual STS task, translated into Spanish.

We apply YiSi-2 directly to these pairs of text fragments, using bilingual word embeddings trained under three different conditions (details of the training sets are given in Table 1):

bivec : BWE's are produced with *bivec*, trained on WMT 2013 ES-EN parallel training data.

vecmap : BWE's are produced with *vecmap*, trained on all WMT 2013 ES and WMT 2019 EN monolingual data, using *Wikititles* as bilingual lexicon.[6]

BERT : BWE's are obtained from pre-trained multilingual BERT models.

We compare the YiSi-2 approach to direct cosine computations on sums of bilingual word embeddings (*bivec_sum*, *vecmap_sum* and *bert_sum*). We also compare our approach to an MT-based approach, in which each Spanish fragment is first machine-translated into English, then compared to the original English fragment, using English word embeddings, produced with *word2vec* trained on WMT 2019 news translation task monolingual data. Similarity is measured either as the cosine of the sums of word vectors from each fragment (*w2v_sum*), or with YiSi using monolingual embeddings as if they were bilingual (YiSi-1_{w2v}).

[5] In this task, the order of languages in pairs was randomized, so that it was first necessary to detect which fragment was in which language. Here, we work from properly ordered pairs.

[6] https://linguatools.org/tools/corpora/wikipedia-parallel-titles-corpora/

Model	Training Data: lang.	domain	#sent	#words	Dictionary #pairs	Embedding vocab #words
bivec	es en	WMT 2013: EU Parliament and web	3.8M	107M 102M	—	291k 220k
vecmap	es en	WMT 2013: News and EU Parliament WMT 2019: News	45M 779M	1B 13B	373k	883k 3M

Table 1: Statistics of data used in training the bilingual word embeddings for evaluating crosslingual lexical semantic similarity in YiSi-2.

SemEval-16 crosslingual STS		
system	news	multisource
MT + monolingual STS		
UWB	**0.9062**	**0.8190**
MT+w2v_sum	0.5883	0.2021
MT+YiSi-1_{w2v}	0.8965	0.6212
crosslingual STS		
bivec_sum	0.5302	0.2684
vecmap_sum	0.3075	0.5398
bert_sum	0.7223	0.6071
YiSi-2_{bivec}	**0.8744**	0.6550
YiSi-2_{vecmap}	0.7854	0.7028
YiSi-2_{bert}	0.8723	**0.7190**

Table 2: Pearson's correlation of the system scores with the gold standard on the two test sets from the SemEval-16 crosslingual STS task.

The MT system used is a phrase-based SMT system, trained using standard resources – Europarl, Common Crawl (CC) and News & Commentary (NC) – totaling approximately 110M words in each language. We bias the SMT decoder to produce a translation that is as close as possible on the surface to the English sentence. This is done by means of log-linear model features that aim at maximizing n-gram precision between the MT output and the English sentence. More details on this method can be found in Lo et al. (2016).

3.2 Results

The results of these experiments are presented in Table 2, where performance is measured in terms of Pearson's correlation with the test sets' gold standard annotations. For reference, we also include results obtained by the UWB system (Brychcín and Svoboda, 2016), which was the best performing system in the SemEval 2016 crosslingual STS shared task. The UWB system is an MT-based system with a STS system trained on assorted lexical, syntactic and semantic features. Globally, using the YiSi metric to measure semantic similarity performs much better than sentence-level cosine ("*_sum" systems). On the *News* dataset, the best results are obtained by combining an MT-based approach with YiSi-1 using monolingual word embeddings (MT+YiSi-1_{w2v}), reflecting the in-domain nature of the text for the MT system. However, this is followed very closely by both the supervised BWE's (YiSi-2_{bivec}) and BERT (YiSi-2_{bert}), which yield very similar results, and clearly outperform semi-supervised BWE's (YiSi-2_{vecmap}). The nature of the *Multisource* translations appears to be quite different from what supervised BWE's and the MT system have been exposed to in training (YiSi-2_{bivec} and MT+YiSi-1_{w2v}), which possibly explains their much poorer performance on this dataset. In contrast, weakly supervised BWE's and BERT behave much more reliably on this data.

Overall, while MT and supervised BWE's seem to work best with YiSi when large quantities of in-domain training data is available, the fully unsupervised alternative of using a pretrained BERT model comes very close, and behaves much better in the face of out-of-domain data.

4 Experiment on Parallel Corpus Filtering

Next, we evaluate YiSi on the task of *Parallel Corpus Filtering* (PCF). Quality – or "cleanliness" – of parallel training data for MT has been shown to affect MT quality at different degrees, and various characteristics of the data – parallelism of the sentence pairs and the grammaticality of target-language data – impact MT systems in different ways (Goutte et al., 2012; Simard, 2014; Khayrallah and Koehn, 2018).

Here, we use data from the WMT19 shared task on PCF. In this shared task, participants were challenged to find good quality translations from noisy sentence-aligned parallel corpora, for the purpose of training MT systems for translating from two low-resource languages, Nepali and Sin-

Model	Training Data: lang.	domain	#sent	#words	Dictionary #pairs	Embedding vocab #words
bivec	ne	IT and religious	563k	8M	—	34k
	en			5M		46k
vecmap	ne	wiki	92k	5M	9k	55k
	en	news	779M	13B		3M

Table 3: Statistics of data used in training the bilingual word embeddings for evaluating crosslingual lexical semantic similarity in YiSi-2.

hala, into English.[7] Both corpora were crawled from the web, using ParaCrawl (Koehn et al., 2018a). Specifically, the task is to produce a score for each sentence pair in these noisy corpora, reflecting the quality of that pair. The scoring schemes are evaluated by extracting the top-scoring sentence pairs from each corpus, then using them to train MT systems; these systems are run on test sets of Wikipedia articles (Guzmán et al., 2019), and the results are evaluated using BLEU (Papineni et al., 2002). In addition to the noisy corpora, participants are allowed to use a few small sets of parallel data, covering different domains, for each of the two low-resource languages, as well as a third, related language, Hindi (which uses the same script as Nepali). The provided data also included much larger monolingual corpora for each of English, Hindi, Nepali and Sinhala.

4.1 Setup

In these experiments, we focus on the Nepali-English corpus, and perform PCF in three steps:

1. **pre-filtering**: apply *ad hoc* filters to remove sentences that are exact duplicates (masking numbers, emails and web addresses), that contain mismatching numbers, that are in the wrong language according to the `pyCLD2` language detector[8] or that are excessively long (either side has more than 150 tokens). We also filter out all pairs where over 50% of the Nepali text is comprised of English, numbers or punctuation.

2. **scoring**: we score sentence pairs using YiSi-2.

3. **re-ranking**: to optimize vocabulary coverage in the resulting MT system, we apply a form of re-ranking: going down the ranked list of scored sentence pairs, we apply a 20% penalty to the pair's score if it does not contain at least one "new" source-language word bigram, i.e., a pair of consecutive source-language tokens not observed in previous (higher-scoring) sentence pairs. This has the effect of down-ranking sentences that are too similar to previously selected sentences.

[7] http://www.statmt.org/wmt19/parallel-corpus-filtering.html

[8] https://github.com/aboSamoor/pycld2

WMT19 parallel corpus filtering		
system	1M-word	5M-word
random	1.30	3.01
Zipporah	**4.14**	**4.42**
YiSi-2$_{bivec}$	3.86	3.76
YiSi-2$_{vecmap}$	4.00	3.76
YiSi-2$_{bert}$	3.77	3.77

Table 4: Uncased BLEU scores on the official WMT19 PCF dev ("dev-test") sets achieved by the SMT systems trained on the 1M- and 5M-word corpora subselected by the scoring systems.

The scoring step is performed with YiSi-2, using bilingual word embeddings obtained under three different conditions (details of the various training sets used can be found in Table 3):

bivec : supervised BWE's produced using *bivec*, trained on the WMT19 PCF (clean) parallel data.

vecmap : weakly supervised BWE's are produced with *vecmap*, trained on all monolingual WMT19 PCF data, using *Wikititles* and the provided dictionary entries as bilingual lexicon.

BERT : BWE's obtained from pretrained multilingual BERT models.

As in the WMT19 PCF shared task, we evaluate the quality of our scoring by training MT systems and measuring their performance on the official test set. We used the provided software to

extract the 1M-word and 5M-word samples from the original test corpora, using the scores of each of our systems in turn. We then trained MT systems using the extracted data: our MT systems are standard phrase-based SMT systems, with components and parameters similar to the German-English SMT system in Williams et al. (2016).

4.2 Results

BLEU scores of the resulting MT systems are shown in Table 4. For comparison, we present the results of random scoring, as well as results obtained by the Zipporah PCF method (Xu and Koehn, 2017). Zipporah combines fluency and adequacy features to score sentence pairs; adequacy features are derived from existing parallel corpora, and the feature combination (logistic regression) is optimized on in-domain parallel data. Therefore, Zipporah can be seen as a fully supervised method. The Zipporah-based MT systems were trained similarly to other systems in the results reported here.

All systems produced with YiSi-2 produce similar results. Interestingly, the MT systems produced with YiSi-2 in the 5M-word condition are not better than those of the 1M-word condition. This is possibly explained by the large quantity of noisy data in the WMT19 Nepalese-English corpus: it is not even clear that there are 5M words of proper translations in that corpus. In such harsh conditions, pre- and post-processing steps become crucially important, and deduplicating the data may even turn out to be harmful, if that means allowing more space for noise. The MT systems produced with Zipporah all achieve higher BLEU scores than YiSi-2, which may be explained by Zipporah's explicit modeling of target-language fluency. This is especially apparent in the 5M-word condition, but it may explain Zipporah's slightly better performance in the 1M-word condition as well. Overall, the benefits of supervised and weakly supervised approaches over using a pre-trained BERT model for PCF appear to be minimal, even in very low-resource conditions such as this.

5 Experiments on Translation Equivalence Error Detection

Given a text and its translation, *Translation Equivalence Error Detection* (TEED) is the task of identifying pairs of corresponding text segments whose meanings are not strictly equivalent. Note that, while in practice "translation errors" can take many forms, here, we are strictly focusing on meaning errors. In this formulation of the problem, we are also assuming that the source and target texts have been properly segmented into sentences and aligned.

The TEED problem is essentially the same as that of Parallel Corpus Filtering (PCF), discussed in the previous section. However, the usage scenario is quite different: in PCF, one is typically dealing with a very large collection of segment pairs, only a fraction of which are true translations; the PCF task is then to filter out pairs which are not proper translations, possibly with some tolerance for pairs of segments that do share partial meaning. In TEED, the data is mostly expected to be high-quality translations; the task is then to identify those pairs that deviate from this norm, even on small details.

5.1 Setup

We experiment the TEED task using a data set obtained from the Canadian government's Public Service Commission (PSC). As part of its mandate, the PSC periodically audits Canadian government job ads, to ensure that they conform with Canada's *Official Languages Act*: as such, job ads must be posted in both of Canada's official languages, English and French, and both versions must be equivalent in meaning.

Our *PSC* data set consists of 175 000 "Statement of merit criteria" paragraphs, identifying any skill, ability, academic specialization, relevant experience or any other essential or asset criteria required for a position to be filled. Of these, 3521 have been manually annotated for equivalence errors by PSC auditors. Out of the 3521 pairs, 164 (4.6%) were reported to contain equivalence errors. The majority of these errors result from missing information in one language or the other (45%). In a slightly smaller proportion (43%), we find pairs of segments that don't express exactly the same meaning – a surprisingly large proportion of this last group consists in cases where the word *and* is translated as *or* or vice-versa. The rest consist in terminology issues and untranslated segments.

We experimented applying the YiSi-2 metric to this task, using bilingual word embeddings obtained under four different conditions:

Model	Training Data: lang.	domain	#sent	#words	Dictionary #pairs	Embedding vocab #words
WMT.bivec	fr	News and EU Parliament	40.7M	1.2B	—	878k
	en			1.4B		791k
PSC.bivec	fr	Job ads	175k	3.0M	—	10k
	en			2.5M		11k
PSC.vecmap	fr	Job ads	175k	3.0M	6k	10k
	en	Job ads	175k	2.5M		11k

Table 5: Statistics of data used in training the bilingual word embeddings for evaluating translation equivalence assessment.

PSC Translation Error Detection

model	ROC AUC	mean F_1	mean F_2
PSC.bivec	**0.807**	**0.160**	**0.281**
PSC.vecmap	0.717	0.136	0.241
BERT	0.702	0.132	0.234
WMT.bivec	0.641	0.112	0.205

Table 6: Sentence-level translation error detection results on PSC test data, expressed in terms of Area under the ROC curve, mean F_1 and mean F_2.

PSC.bivec : BWE's are produced with *bivec*, trained on all unannotated PSC data.

PSC.vecmap : BWE's are produced with *vecmap*, trained on all unannotated PSC data, using *wikititles* as bilingual lexicon.

WMT.bivec : BWE's are produced with *bivec*, trained on all bilingual French-English data provided for the WMT 2015 News translation shared task.

BERT : BWE's obtained from pretrained multilingual BERT models.

Details about the training data can be found in Table 5.

5.2 Results

For these experiments, we considered an application scenario in which a text and its translation, in the form of pairs of matching segments, are scored using YiSi-2, and presented to a user, ranked in increasing order of score, so that pairs most likely to contain a translation error are presented first. The performance of the system can then be measured in terms of true and false positive rates, precision and recall, over subsets of increasing sizes of the test set. In Table 6, we report results in terms of mean F-score, with $\beta = 1$ and $\beta = 2$, and in terms

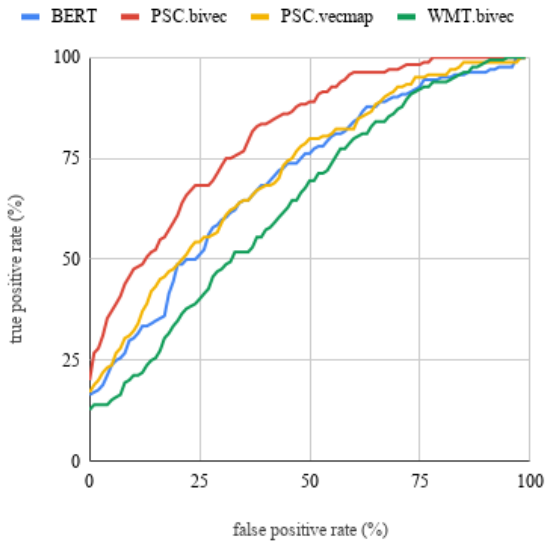


Figure 1: ROC curves of Sentence-level translation error detection results on PSC test data.

of the *Area under the ROC curve* (ROC AUC), which can be interpreted as the probability that a system will score a randomly chosen faulty translation lower than a randomly chosen good translation. The ROC curves themselves can be seen in Figure 1.

Globally, YiSi-2 clearly performs best at this task when using BWE's trained on domain-specific parallel data (PSC.bivec), even when there is very limited quantities of such data, as is the case here. However, BERT models perform comparably to vector-mapped BWE's trained with in-domain data (PSC.vecmap), and substantially better than BWE's trained on large quantities of generic, out-of-domain parallel data (WMT). We conclude that, in the absence of in-domain parallel data, for TEED applications, an unsupervised YiSi-2 method will perform at least as well as supervised methods trained on out-of-domain data.

6 Conclusion

We presented a fully unsupervised crosslingual semantic textual similarity (STS) metric, based on contextual embeddings extracted from BERT without fine-tuning. We perform intrinsic evaluations on crosslingual STS data sets and extrinsic evaluations on parallel corpus filtering and human translation equivalence assessment tasks. Our results show that the unsupervised metric we propose achieves performance on par with supervised or weakly supervised approaches. We show that the circular dependency on the existence of parallel resources for using crosslingual STS to identify parallel data can be broken.

In this paper, we have only experimented with the contextual embeddings extracted from pre-trained multilingual BERT model. For domain-specific applications, such as the job advertisement domain in the PSC translation equivalence error detection task, the performance of YiSi-2 could potentially be improved by fine-tuning BERT with in-domain data, something we plan to examine in the near future. We will also want to explore the use of other multilingual context representation models, such as MUSE (Conneau et al., 2017), XLM (Lample and Conneau, 2019), etc.

Acknowledgement

We would like to thank Dominic Demers and his colleagues at the Canadian governments Public Service Commission (PSC) for providing the PSC data set and managing the human annotation in the experiments.

References

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016a. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California. Association for Computational Linguistics.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016b. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, San Diego, California. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294.

Wilker Aziz and Lucia Specia. 2011. Fully automatic compilation of Portuguese-English and Portuguese-Spanish parallel corpora. In *Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology*.

Tomáš Brychcín and Lukáš Svoboda. 2016. UWB at SemEval-2016 task 1: Semantic textual similarity using lexical, syntactic, and semantic information. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 588–594, San Diego, California. Association for Computational Linguistics.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, EMSEE '05, pages 13–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Marc Franco-Salvador, Parth Gupta, Paolo Rosso, and Rafael E. Banchs. 2016a. Cross-language plagiarism detection over continuous-space- and knowledge graph-based representations of language. *Know.-Based Syst.*, 111(C):87–99.

Marc Franco-Salvador, Paolo Rosso, and Manuel Montes-y Gómez. 2016b. A systematic study of knowledge graph analysis for cross-language plagiarism detection. *Inf. Process. Manage.*, 52(4):550–570.

Marc Franco-Salvador, Paolo Rosso, and Roberto Navigli. 2014. A knowledge-based representation for cross-language document retrieval and categorization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 414–423, Gothenburg, Sweden. Association for Computational Linguistics.

Cyril Goutte, Marine Carpuat, and George Foster. 2012. The impact of sentence alignment errors on phrase-based machine translation performance. In *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas*.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. Two new evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english. *CoRR*, abs/1902.01382.

Huda Khayrallah and Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. *CoRR*, abs/1805.12282.

Philipp Koehn, Kenneth Heafield, Mikel L. Forcada, Miquel Esplà-Gomis, Sergio Ortiz-Rojas, Gema Ramírez Sánchez, Víctor M. Sánchez Cartagena, Barry Haddow, Marta Bañón, Marek Střelec, Anna Samiotou, and Amir Kamran. 2018a. ParaCrawl corpus version 1.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Philipp Koehn, Huda Khayrallah, Kenneth Heafield, and Mikel Forcada. 2018b. Findings of the wmt 2018 shared task on parallel corpus filtering. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, Brussels, Belgium. Association for Computational Linguistics.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Chi-kiu Lo. 2019. YiSi - a unified semantic mt quality evaluation and estimation metric for languages with different levels of available resources. In *Proceedings of the Fourth Conference on Machine Translation*, pages 706–712, Florence, Italy. Association for Computational Linguistics.

Chi-kiu Lo, Cyril Goutte, and Michel Simard. 2016. CNRC at SemEval-2016 task 1: Experiments in crosslingual semantic textual similarity. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 668–673, San Diego, California. Association for Computational Linguistics.

Chi-kiu Lo, Michel Simard, Darlene Stewart, Samuel Larkin, Cyril Goutte, and Patrick Littell. 2018. Accurate semantic textual similarity for cleaning noisy parallel corpora using semantic machine translation evaluation metric: The NRC supervised submissions to the parallel corpus filtering task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 908–916, Belgium, Brussels. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *NAACL Workshop on Vector Space Modeling for NLP*, Denver, United States.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Matteo Negri, Alessandro Marchetti, Yashar Mehdad, Luisa Bentivogli, and Danilo Giampiccolo. 2013. Semeval-2013 task 8: Cross-lingual textual entailment for content synchronization. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 25–33, Atlanta, Georgia, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? *CoRR*, abs/1906.01502.

Philip Resnik and Noah A. Smith. 2003. The web as a parallel corpus. *Comput. Linguist.*, 29(3):349–380.

Michel Simard. 2014. Clean data for training statistical MT: the case of MT contamination. In *Proceedings of the Eleventh Conference of the Association for Machine Translation in the Americas*, pages 69–82, Vancouver, BC, Canada.

Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramón Astudillo, and André F. T. Martins. 2018. Findings of the wmt 2018 shared task on quality estimation. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 702–722, Belgium, Brussels. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz

Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 363–372, New York, NY, USA. ACM.

Philip Williams, Rico Sennrich, Maria Nadejde, Matthias Huck, Barry Haddow, and Ondřej Bojar. 2016. Edinburgh's statistical machine translation systems for wmt16. In *Proceedings of the First Conference on Machine Translation*, pages 399–410, Berlin, Germany. Association for Computational Linguistics.

Hainan Xu and Philipp Koehn. 2017. Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2945–2950, Copenhagen, Denmark. Association for Computational Linguistics.

On the Importance of Subword Information for Morphological Tasks in Truly Low-Resource Languages

Yi Zhu[1]*, **Benjamin Heinzerling**[2,3]*, **Ivan Vulić**[1],
Michael Strube[4], **Roi Reichart**[5], **Anna Korhonen**[1]
[1]Language Technology Lab, University of Cambridge
[2]RIKEN AIP, [3]Tohoku University
[4]Heidelberg Institute for Theoretical Studies, [5]Technion, IIT
{yz568,iv250,alk23}@cam.ac.uk, benjamin.heinzerling@riken.jp
michael.strube@h-its.org, roiri@technion.ac.il

Abstract

Recent work has validated the importance of subword information for word representation learning. Since subwords increase parameter sharing ability in neural models, their value should be even more pronounced in low-data regimes. In this work, we therefore provide a comprehensive analysis focused on the usefulness of subwords for word representation learning in truly low-resource scenarios and for three representative morphological tasks: fine-grained entity typing, morphological tagging, and named entity recognition. We conduct a systematic study that spans several dimensions of comparison: **1)** *type of data scarcity* which can stem from the lack of task-specific training data, or even from the lack of unannotated data required to train word embeddings, or both; **2)** *language type* by working with a sample of 16 typologically diverse languages including some truly low-resource ones (e.g. Rusyn, Buryat, and Zulu); **3)** the choice of the *subword-informed word representation method*. Our main results show that subword-informed models are universally useful across all language types, with large gains over subword-agnostic embeddings. They also suggest that the effective use of subwords largely depends on the language (type) and the task at hand, as well as on the amount of available data for training the embeddings and task-based models, where having sufficient in-task data is a more critical requirement.

1 Introduction and Motivation

Recent studies have confirmed the usefulness of leveraging subword-level information in learning word representations (Peters et al., 2018; Heinzerling and Strube, 2018; Grave et al., 2018; Zhu et al., 2019, *inter alia*), and in a range of tasks such as sequence tagging (Lample et al., 2016; Akbik et al., 2018; Devlin et al., 2019), fine-grained entity typing (Zhu et al., 2019), neural machine translation (Sennrich et al., 2016; Luong and Manning, 2016; Lample et al., 2018; Durrani et al., 2019), or general and rare word similarity (Pilehvar et al., 2018; Zhu et al., 2019). The subword-informed word representation architectures leverage the internal structure of words and assume that a word's meaning can be inferred from the meaning of its constituent (i.e., subword) parts. Instead of treating each word as an atomic unit, subword-informed neural architectures reduce data sparsity by relying on parameterization at the level of subwords (Bojanowski et al., 2017; Pinter et al., 2017; Chaudhary et al., 2018; Kudo, 2018).

An increasing body of work focuses on various aspects of subword-informed representation learning such as segmentation of words into subwords and composing subword embeddings into word representations (Lazaridou et al., 2013; Cotterell and Schütze, 2015, 2018; Avraham and Goldberg, 2017; Vania and Lopez, 2017; Kim et al., 2018; Zhang et al., 2018; Zhao et al., 2018, *inter alia*).[1] The increased parameter sharing ability of such models is especially relevant for learning embeddings of rare and unseen words. Therefore, the importance of subword-level knowledge should be even more pronounced in *low-data regimes* for *truly low-resource languages*. Yet, a systematic study focusing exactly on the usefulness of subword information in such settings is currently missing in the literature. In this work, we fill this gap by providing a comprehensive analysis of subword-informed representation learning focused on low-resource setups.

Our study centers on the following axes of comparison, focusing on three representative tasks where subword-level information can guide learn-

*Equal contribution, work partly done while at HITS.

[1]An overview of a variety of subword-informed word representation architectures and different segmentation and composition strategies is provided by Zhu et al. (2019).

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 216–226
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

ing, namely fine-grained entity typing (FGET), morphological tagging (MTAG), and named entity recognition (NER): **1)** Since data scarcity can stem from unavailability of (i) task-specific training data or (ii) unannotated corpora to train the embeddings in the first place, or (iii) both, we analyse how different data regimes affect the final task performance. **2)** We experiment with 16 languages representing 4 diverse morphological types, with a focus on truly low-resource languages such as Zulu, Rusyn, Buryat, or Bambara. **3)** We experiment with a variety of subword-informed representation architectures, where the focus is on unsupervised, widely portable language-agnostic methods such as the ones based on character n-grams (Luong and Manning, 2016; Bojanowski et al., 2017), Byte Pair Encodings (BPE) (Sennrich et al., 2016; Heinzerling and Strube, 2018), Morfessor (Smit et al., 2014), or BERT-style pretraining and fine-tuning (Devlin et al., 2019) which relies on WordPieces (Wu et al., 2016). We demonstrate that by tuning subword-informed models in low-resource settings we can obtain substantial gains over subword-agnostic models such as skip-gram with negative sampling (Mikolov et al., 2013) across the board.

The main goal of this study is to identify viable and effective subword-informed approaches for truly low-resource languages and offer modeling guidance in relation to the target task, the language at hand, and the (un)availability of general and/or task-specific training data. As expected, our key results indicate that there is no straightforward *"one-size-fits-all"* solution, although certain approaches (e.g., BPE-based or character n-grams) emerge as more robust in general. The optimal subword-informed configurations are largely task-, language-, and resource-dependent: their performance hinges on a complex interplay of the multiple factors mentioned above. For instance, we show that fine-tuning pretrained multilingual BERT (Devlin et al., 2019; Wu and Dredze, 2019) is a viable strategy for "double" low-resource settings in the NER and MTAG tasks, but it fails for the FGET task in the same setting; furthermore, its performance can be matched or surpassed by other subword-informed methods in NER and MTAG as soon as they obtain sufficient embedding training data.

2 Methodology

In what follows, we further motivate our work by analyzing two different sources of data scarcity: embedding training data (termed *WE data*) and task-specific training data (termed *task data*). Following that, we motivate our selection of test languages and outline the subword-informed representation methods compared in our evaluation.

Types of Data Scarcity. The majority of languages in the world still lack basic language technology, and progress in natural language processing is largely hindered by the lack of annotated *task data* that can guide machine learning models (Agić et al., 2016; Ponti et al., 2018). However, many languages face another challenge: the lack of *large unannotated text corpora* that can be used to induce useful general features such as word embeddings (Adams et al., 2017; Fang and Cohn, 2017; Ponti et al., 2018):[2] i.e. WE data.

The absence of data has over the recent years materialized the *proxy fallacy*. That is, methods tailored for low-resource languages are typically tested only by proxy, simulating low-data regimes exclusively on resource-rich languages (Agić et al., 2017). While this type of evaluation is useful for analyzing the main properties of the intended low-resource methods in controlled *in vitro* conditions, a complete evaluation should also provide results on true low-resource languages *in vivo*. In this paper we therefore conduct both types of evaluation. Note that in this work we still focus on low-resource languages that have at least some digital footprint (see the statistics later in Table 1), while handling zero-resource languages without any available data (Kornai, 2013; Ponti et al., 2018) is a challenge left for future work.

(Selection of) Languages. Both *sources of data scarcity* potentially manifest in degraded task performance for low-resource languages: our goal is to analyze the extent to which these factors affect downstream tasks across morphologically diverse language types that naturally come with varying data sizes to train their respective embeddings and task-based models. Our selection of test languages is therefore guided by the following goals: **a)** following recent initiatives (e.g. in language modeling) (Cotterell et al., 2018; Gerz et al., 2018), we aim to ensure coverage of different genealogical and typological properties; **b)** we aim to cover low-resource languages with varying amounts of available WE data and task-specific data.

We select 16 languages in total spanning 4 broad

[2]For instance, as of April 2019, Wikipedia is available only in 304 out of the estimated 7,000 existing languages.

	Agglutinative						Fusional						Introflexive		Isolating	
	BM	BXR	MYV	TE	TR	ZU	EN	FO	GA	GOT	MT	RUE	AM	HE	YO	ZH
EMB	40K	372K	207K	5M	5M	69K	5M	1.6M	4.4M	18K	1.5M	282K	659K	5M	542K	5M
FGET	29K	760	740	13K	60K	36K	60K	30K	56K	289	2.7K	1.5K	2.2K	60K	15K	60K
NER	345	2.4K	2.1K	9.9K	167K	425	8.9M	4.0K	7.6K	475	1.9K	1.6K	1.0K	107K	3.4K	–
MTAG	–	–	–	1.1K	3.7K	–	24K	–	–	3.4K	1.1K	–	–	5.2K	–	4.0K
BERT				✓	✓		✓		✓					✓	✓	✓

Table 1: Overview of test languages and data availability. EMB denotes the maximum number of tokens in corresponding Wikipedias used for training embeddings. Actual Wikipedia sizes are larger than 5M for (TE, TR, EN, HE, ZH), but were limited to 5M tokens in order to ensure comparable evaluation settings for data scarcity simulation experiments across different languages. FGET, NER, and MTAG rows show the number of instances for the three evaluation tasks (see §3): number of entity mentions for FGET, number of sentences for NER and MTAG. In MTAG, we omit languages for which UDv2.3 provides only a test set, but no training set. The BERT row shows the languages supported by multilingual BERT. Languages are identified by their ISO 639-1 code.

morphological types, listed in Table 1. Among these, we chose one (relatively) high-resource language for each type: Turkish (agglutinative), English (fusional), Hebrew (introflexive), and Chinese (isolating). We use these four languages to *simulate* data scarcity scenarios and run experiments where we *control* the degree of data scarcity related to both embedding training data and task-related data. The remaining 12 languages are treated as test languages with varying amounts of available data (see Table 1. For instance, relying on the Wikipedia data for embedding training, Gothic (GOT) is the language from our set that contains the fewest number of word tokens in its respective Wikipedia (18K, in terms of Wikipedia size this ranks it as 273th out of 304 Wikipedia languages); Irish Gaelic (GA) with 4.4M tokens is ranked $87/304$.

Subword-Informed Word Representations. We mainly follow the framework of Zhu et al. (2019) for the construction of subword-informed word representations; the reader is encouraged to refer to the original paper for more details. In short, to compute the representation for a given word $w \in V$, where V is the word vocabulary, the framework is based on three main components: 1) *segmentation* of words into subwords, 2) *interaction* between subword and position embeddings, and 3) a *composition function* that yields the final word embedding from the constituent subwords. Zhu et al. (2019) explored a large space of possible subword-informed configurations. Based on their findings, we select a representative subset of model configurations. They can be obtained by varying the components listed in Table 2.

Concretely, w is first segmented into an ordered subword sequence from the subword vocabulary S by a deterministic subword segmentation method. To enable automatic language-agnostic segmentation across multiple languages, we focus on unsupervised segmentation methods: we work with Morfessor (Smit et al., 2014), character n-grams (Bojanowski et al., 2017) and BPE (Gage, 1994). We use the default parameters for Morfessor, and the same 3 to 6 character n-gram range as Bojanowski et al. (2017). For BPE, the number of merge operations is a tunable hyper-parameter. It controls the segmentation "aggressiveness": the larger the number the more conservative the BPE segmentation is. Following Heinzerling and Strube (2018), we investigate the values $\{1e3, 1e4, 1e5\}$: this allows us to test varying segmentation granularity in relation to different language types.

After segmentation into subwords, each subword is represented by a vector $\mathbf{s}$ from the subword embedding matrix $\mathbf{S} \in \mathbb{R}^{|S| \times d}$, where d is the dimensionality of subword embeddings. Optionally, the word itself can be appended to the subword sequence and embedded into the subword space in order to incorporate word-level information (Bojanowski et al., 2017). To encode subword order, $\mathbf{s}$ can be further enriched by a trainable position embedding $\mathbf{p}$. We use addition to combine subword and position embeddings, namely $\mathbf{s} := \mathbf{s} + \mathbf{p}$, which has become the de-facto standard method to encode positional information (Gehring et al., 2017; Vaswani et al., 2017; Devlin et al., 2019).

Finally, the subword embedding sequence is passed to a composition function, which computes the final word representation. Li et al. (2018) and Zhu et al. (2019) have empirically verified that composition by simple addition, among other more complex composition functions, is a robust choice. Therefore, we use addition in all our experiments.

Similar to Bojanowski et al. (2017); Zhu et al. (2019), we adopt skip-gram with negative sampling

Component	Option	Label
Segmentation	Morfessor	`morf`
	BPE	`bpeX`
	char n-gram	`charn`
Word token	exclusion	*w-*
	inclusion	*w+*
Position embedding	exclusion	*p-*
	additive	*p+*
Composition function	addition	*add*

Table 2: Components for constructing subword-informed word representations. In the `bpeX` label $X \in \{1e3, 1e4, 1e5\}$ denotes the BPE vocabulary size.

(Mikolov et al., 2013) as our word-level distributional model: the target word embedding is computed by our subword-informed model, and the context word is parameterized by the (word-level) context embedding matrix $\mathbf{W}_c \in \mathbb{R}^{|V| \times d_c}$.

We compare subword-informed architectures to two well-known word representation models, also captured by the general framework of Zhu et al. (2019): 1) the subword-agnostic skip-gram model from the `word2vec` package (Mikolov et al., 2013) (W2V), and 2) `fastText` (FT) (Bojanowski et al., 2017). The comparison to W2V aims to validate the potential benefit of subword-informed word representations for truly low-resource languages, while the comparison to FT measures the gains that can be achieved by more sophisticated and fine-tuned subword-informed architectures. We also compare with pretrained multilingual $BERT_{base}$ (Devlin et al., 2019) on the languages supported by this model.

3 Evaluation Tasks

Fine-Grained Entity Typing. FGET is cast as a sequence classification problem, where an entity mention consisting of one or more tokens (e.g. *Lincolnshire*, *Bill Clinton*), is mapped to one of the 112 fine-grained entity types from the FIGER inventory (Ling and Weld, 2012; Yaghoobzadeh and Schütze, 2015; Heinzerling and Strube, 2018). Since entity mentions are short token sequences and not full sentences, this semi-morphological/semantic task requires a model to rely on the subword information of individual tokens in the absence of sentence context. That is, subwords can provide evidence useful for entity type classification in the absence of context. For instance, *Lincolnshire* is assigned the type `/location/county` as *-shire* is a suffix that strongly indicates a location. Hence, FGET is well-suited for evaluating subword-informed representations, and can benefit from the information.

Morphological Tagging. MTAG is the task of annotating each word in a sentence with features such as part-of-speech, gender, number, tense, and case. These features are represented as a set of key-value pairs. For example, *classified* is a finite (Fin) verb (V) in indicative (Ind) mood, third person, past tense, which is annotated with the morphological tag {POS=V, Mood=Ind, Person=3, Tense=Past, VerbForm=Fin}, and the female singular third-person possessive personal pronoun *her* with the morphological tag {Gender=Fem, Number=Sing, Person=3, Poss=Yes, PronType=Prs}.

Named Entity Recognition. NER is the task of annotating textual mentions of real-world entities with their semantic type, such as person, location, and organization: e.g., *Barack Obama (person) was born in Hawaii (location).*

4 Experimental Setup

Embedding Training: WE Data. For training word and subword embeddings, we rely on Wikipedia text for all 16 languages, with corresponding Wikipedia sizes listed in Table 1. For training embeddings in controlled low-resource settings with our 4 "simulation" languages, we sample nine data points to simulate low-resource scenarios with WE data. Specifically, we sample 10K, 20K, 50K, 100K, 200K, 500K, 1M, 2M, and 5M tokens of article text for each of the 4 languages. For the other 12 languages we report results obtained by training embedding on the full Wikipedia edition.

Task-Specific Data: Task Data. The maximum number of training instances for all languages is again provided in Table 1. As before, for 4 languages we simulate low-resource settings by taking only a sample of the available task data: for FGET we work with 200, 2K or 20K training instances which roughly correspond to training regimes of different data availability, while we select 300,[3] 1K, and 10K sentences for NER and MGET. Again, for the remaining 12 languages, we use all the available data to run the experiments. We adopt existing data splits into training, development, and test portions for MTAG (Cotterell and Heigold, 2017), and random splits for FGET (Heinzerling and Strube, 2018; Zhu et al., 2019) and NER (Pan et al., 2017).

[3] With a smaller number of instances (e.g., 100), NER and MGET model training was unstable and resulted in near-zero performance across multiple runs.

A large number of data points for scarcity simulations allow us to trace how performance on the three tasks varies in relation to the availability of WE data versus task data, and what data source is more important for the final performance.

Embedding Training Setup. When training our subword-informed representations, we argue that keeping hyper-parameters fixed across different data points will possibly result in underfitting for larger data sizes or overfitting for smaller data sizes. Therefore, we split data points into three groups: $[10K, 50K]$ (G1), $(50K, 500K]$ (G2) and $(500K, 5M]$ (G3), and use the same hyper-parameters for word embedding training within the same group. For G1, we train with batch size 32 for 60 epochs and set the minimum word frequency threshold to 2. For G2 the values are: $128/30/3$, and $512/15/5$ for G3. This way, we ensure that 1) the difference of the absolute data sizes can be compared within the same data group, and 2) for the corresponding data points in different groups ($10K$, $100K$, $1M$) the sample efficiency can be compared, as the models trained on these data points undergo roughly the same number of updates.[4]

Task-Specific Training Setup. For FGET, we use the dataset of Heinzerling and Strube (2018) obtained by mapping entity mentions from Wikidata (Vrandečić and Krötzsch, 2014) to their associated FIGER-based most notable entity type (Ling and Weld, 2012). For each language, we randomly sample up to 100k pairs of entity mentions with corresponding entity type and create random 60/20/20 train/dev/test splits. Our FGET model is designed after the hierarchical architecture by Zhu et al. (2019). For each entity token, we first use our subword-informed model to obtain word representations, and then feed the token embedding sequence into a bidirectional LSTM with 2 hidden layers of size 512, followed by a projection layer which predicts the entity type. We initialize our FGET model with the pretrained subword model, and fine-tune it during training. With BERT, we input the entire entity mention and then use the representation of the special `[CLS]` token for classification. We train with early stopping, using Adam (Kingma and Ba, 2015) with default parameters across all languages. As suggested by Wu and Dredze (2019), BERT hyper-parameters are more sensitive to smaller data sizes, so we tune them on the smallest data point with 200 training instances. We follow Wu and Dredze (2019) to select hyper-parameter candidates, i.e., $2e{-}5/3e{-}5/5e{-}5$ for learning rate, $16/32$ for batch size and triangular learning rate scheduler with first 10% of batches as warm-up. We do an exhaustive search on four high resource languages: EN, TR, HE, ZH and select the hyper-parameter combination with the best average score on the development sets.

For MTAG, we evaluate on the multilingual morphological annotations provided by the Universal Dependencies project (Nivre et al., 2016) and adopt the experimental protocol of Cotterell and Heigold (2017). Specifically, we cast MTAG as a sequence labeling task by treating the concatenation of all key-value pairs for a given word as the word's label. As sequence labeling model, we train a bidirectional LSTM (Hochreiter and Schmidhuber, 1997; Plank et al., 2016), with two layers of size 1024 and dropout 0.4, using early stopping on the development set. For experiments involving multilingual BERT, we fine-tune all of BERT's layers and feed the final layer into an LSTM before classification. The evaluation metric is per-label accuracy, i.e., a word's morphological tag is either predicted correctly or not, and there is no partial credit for the correct prediction of only a subset of features.

We evaluate NER performance on WikiAnn (Pan et al., 2017), a multilingual dataset which provides three-class entity type annotations which were automatically extracted from Wikipedia. We train sequence labeling models using exactly the same architectures and hyper-parameters as in MTAG, and report F1 scores. As WikiAnn does not come with predefined train/dev/test sets, we create random 60/20/20 splits.

5 Results and Discussion

Results for data scarcity simulation experiments are summarized in Figures 1-3, while the results on the remaining 12 languages for all three tasks are provided in Tables 3-4, with the best results among different configurations of subword-informed methods reported. As the first main finding, the results show that subword-informed architectures substantially outperform the subword-agnostic skip-gram W2V baseline, and the gaps are in some cases very large: e.g., see the results in Figures 1-3 for the settings with extremely scarce WE data. These scores verify the importance of subword-level knowledge

[4] We train `fastText` and skip-gram from `word2vec` with the same number of epochs that is used to train our subword-informed models on the corresponding data points.

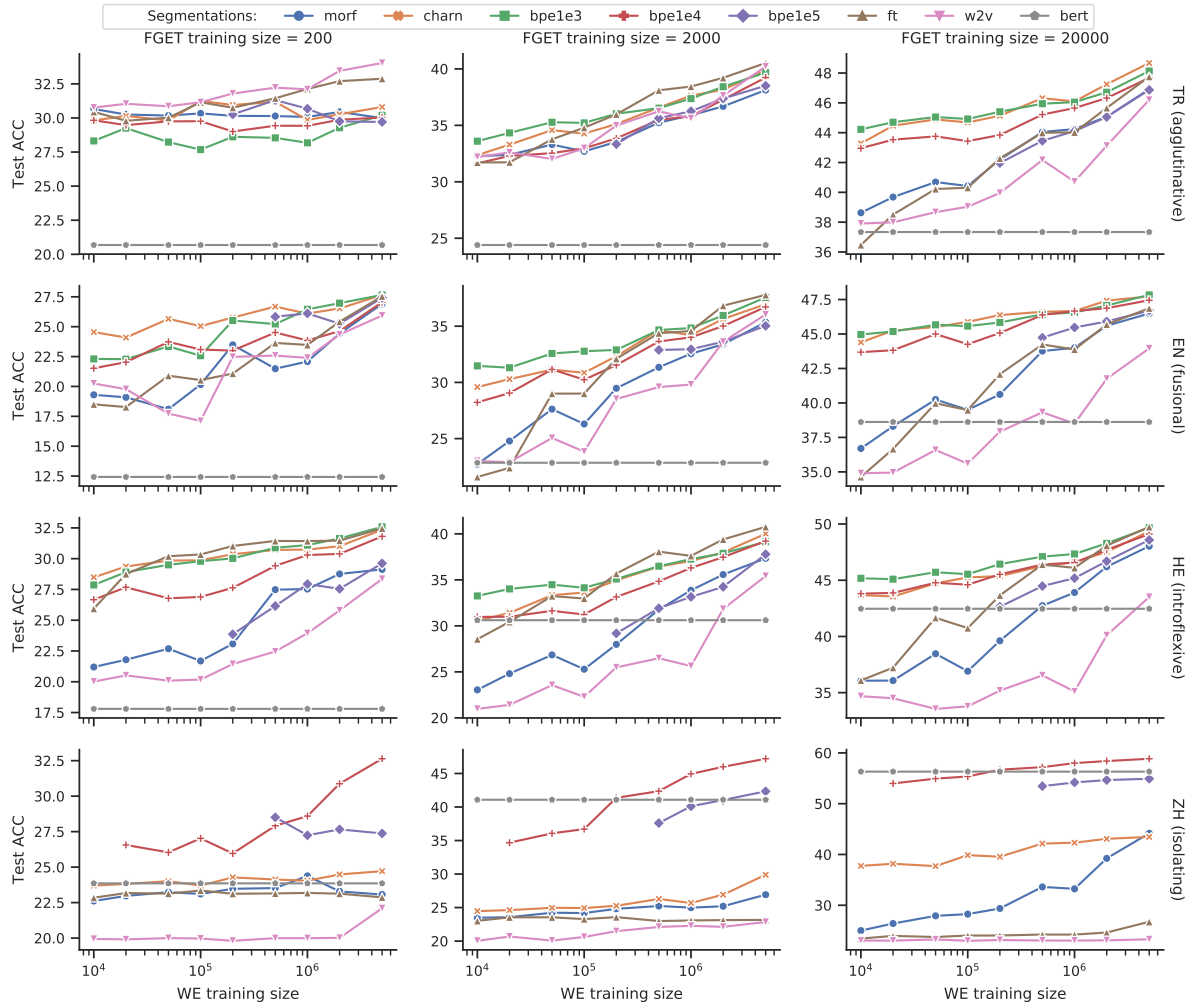


Figure 1: Test performance (accuracy) in the FGET task for different segmentation methods in data scarcity simulation experiments across 4 languages representing 4 broad morphological types, averaged over 5 runs. Some data points with Chinese (ZH) are not shown as in those cases the subword model is reduced to single characters only.

for low-resource setups.

Simulating Data Scarcity. Another general finding concerns the importance of WE data versus task data. The simulation results in Figure 1-3 suggest that both types of data are instrumental to improved task performance: This finding is universal as we observe similar behaviors across tasks and across different languages. While WE data is important, considerably larger gains are achieved by collecting more task data: e.g., see the large gains in FGET when training on 200 versus 2K entity mentions. In summary, both types of data scarcity decrease performance, but the impact of scarce task data seems more pronounced. Collecting more WE data when dealing with scarce task data leads to larger gains in the FGET task compared to MTAG or NER.

While subword models are generally better than the baselines across different data points, less aggressive segmentation models and token-based models close the gap very quickly when increasing WE data, which is in line with the findings of Zhu et al. (2019), where `morf` eventually prevails in this task with abundant WE data. This again verifies the usefulness of subword-level knowledge for low-(WE) data regimes. Similar trends emerge in terms of task data, but the advantage of subword models seems more salient with more task data. The underlying task architectures start making use of subword features more effectively: this shows that subword-level knowledge is particularly useful for the three chosen morphological tasks.

Performance of BERT. An interesting analysis regarding the (relative) performance of pretrained multilingual BERT model emerges from Figures 1-3. Fine-tuned BERT displays much stronger performance in low-resources settings for the MTAG and NER tasks than for the FGET task (e.g., compare the sub-figures in the first columns of the

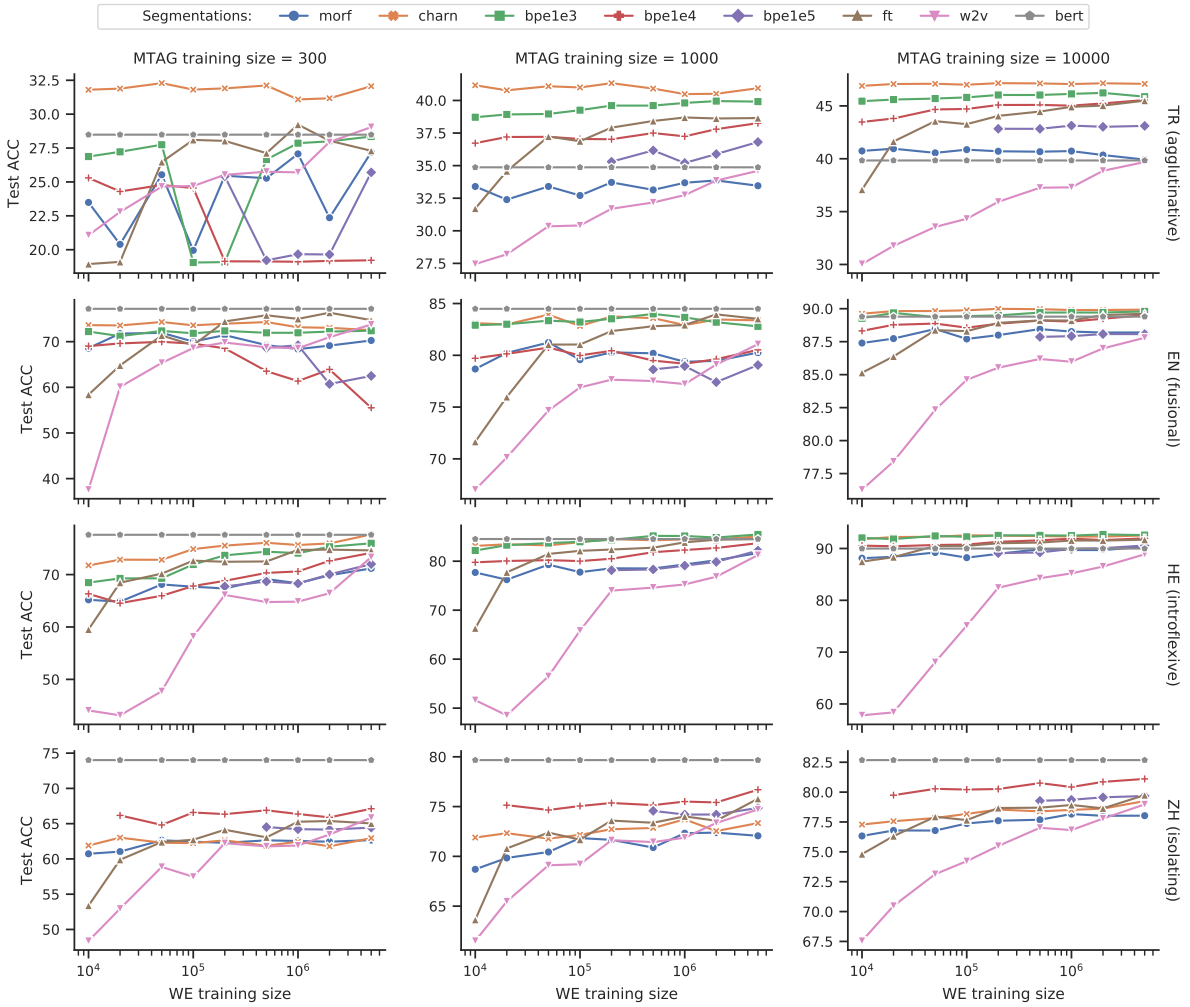


Figure 2: Test performance in the MTAG task in data scarcity simulation experiments.

corresponding figures). The explanation of MTAG and NER performance is intuitive. A pretrained BERT model encodes a massive amount of background knowledge available during its (multilingual) pretraining. However, as we supplement other subword-informed representation learning methods with more data for training the embeddings, the gap gets smaller until it almost completely vanishes: other methods now also get to see some of the distributional information which BERT already consumed in its pretraining.

BERT performance on FGET versus MTAG and NER concerns the very nature of the tasks at hand. The input data for FGET consist mostly of 2-4 word tokens (i.e., entity mentions), while MTAG and NER operate on full sentences as input. Since BERT has been pretrained on sentences, this setting is a natural fit and makes fine-tuning to these tasks easier: BERT already provides a sort of "contextual subword composition function". This stands in contrast with the other subword-informed approaches. There, we might have good non-contextual subword embeddings and a pretrained "non-contextual" composition function, but we have to learn how to effectively leverage the context for the task at hand (i.e., by running an LSTM over the subword-informed token representations) from scratch.[5]

Truly Low-Resource Languages. The results on the 12 test languages in Table 3-4 suggest that subword-informed models are better than the baselines in most cases: this validates the initial findings from the simulation experiments. That is, leveraging subword information is important for WE induction as well as for task-specific training. The

[5]Another factor at play is multilingual BERT's limited vocabulary size (100K WordPiece symbols), leaving on average a bit under 1K symbols per language. Due to the different sizes of Wikipedias used for pretraining BERT, some languages might even be represented with far fewer than 1K vocabulary entries, thereby limiting the effective language-specific model capacity. Therefore, it is not that surprising that monolingual subword-informed representations gradually surpass BERT as more language-specific WE data becomes available. This finding is also supported by the results reported in Table 3.

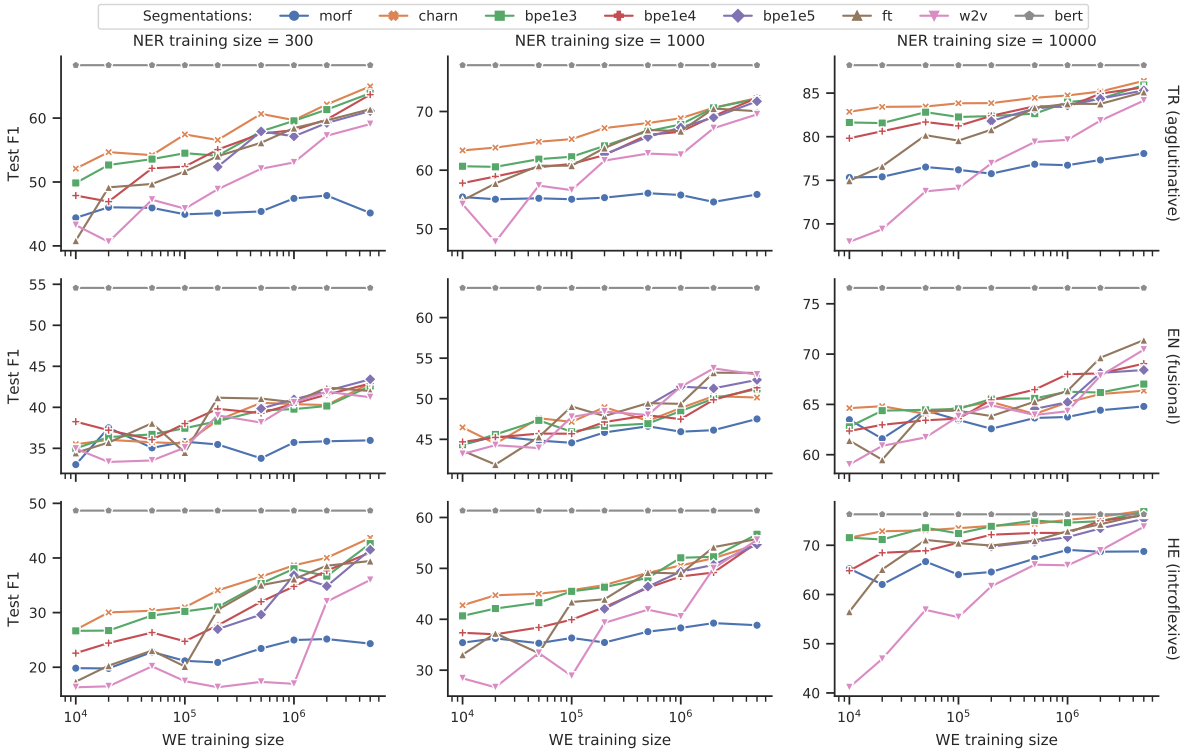


Figure 3: Test performance (F1 score) in the NER task in data scarcity simulation experiments. We do not show results for Chinese as the annotations of the Chinese NER are provided only on the character-level and thus impede experimentation with most of the subword-informed methods used in our evaluation.

gains with subword methods become larger for languages with fewer WE data (e.g., ZU, BM, GOT); this is again consistent with the previously reported simulation experiments.

Tasks, Language Types, Subword Configurations. The results further suggest that the optimal configuration indeed varies across different tasks and language types, and therefore it is required to carefully tune the configuration to reach improved performance. For instance, as agglutinative languages have different granularities of morphological complexity, it is not even possible to isolate a single optimal segmentation method within this language type. Overall, the segmentation based `charn` followed by BPE emerge as most robust choices across all languages and tasks. However, `charn` has the largest number of parameters and is slower to train compared to other segmentations, and in case of BPE its number of merge operations must be tuned to yield competitive scores.

While we do not see extremely clear patterns from the results in relation to particular language types, the scores suggest that for agglutinative and fusional languages a hybrid segmentation such as `charn` or a moderate one (`bpe1e4`, `bpe1e5`) is a good choice. For introflexive and isolating languages, more aggressive segmentations seem to be also competitive in FGET and MTAG, while `bpe1e4` being very effective for ZH, and `charn` (again) and `bpe1e5` seems to be preferred in NER.

Apart from segmentation methods, we also analyzed the effect of word token embeddings ($w+$) and position embeddings ($p+$) in the subword-informed learning framework (Zhu et al., 2019) (see before Table 2 in §2), shown in Figure 4. NER can clearly benefit from both $w+$ and $p+$ and $w+$ is also useful for MTAG. However, for other tasks, the fluctuations between configurations are minimal once the segmentation has been fixed, which suggests that the most critical component is indeed the chosen segmentation method: this is why we have mostly focused on the analyses of the segmentation method and its impact on task performance in this work. Regarding the composition functions, as demonstrated in Zhu et al. (2019), more complex composition functions do not necessarily yield superior results in a range of downstream tasks. We therefore leave the exploration of more sophisticated composition functions for future work.

6 Conclusions and Future Work

We have presented an empirical study focused on the importance of subword-informed word repre-

		Agglutinative					Fusional					Intro	Isolat
		BM	BXR	MYV	TE	ZU	FO	GA	GOT	MT	RUE	AM	YO
FGET	morf	52.43	52.47	79.11	57.79	53.00	54.43	50.77	29.90	49.48	50.38	41.82	83.43
	charn	**56.09**	57.33	81.69	58.83	**56.34**	**58.44**	**52.62**	**34.02**	**54.46**	**58.59**	45.65	84.85
	bpe1e3	53.61	51.30	81.13	58.73	55.41	56.04	50.74	31.55	52.79	55.57	**47.99**	**85.22**
	bpe1e4	54.20	53.81	**81.93**	**59.24**	55.67	56.67	51.47	26.39	52.15	54.81	47.05	84.42
	bpe1e5	-	53.80	80.00	58.13	-	56.31	51.52	-	51.52	52.52	44.74	83.39
	ft	51.91	**57.96**	81.05	57.79	52.62	53.74	49.67	31.96	53.95	53.64	44.80	83.71
	w2v	52.28	42.19	76.86	56.99	52.95	53.07	49.07	24.53	46.61	47.36	36.81	82.56
	bert	-	-	-	49.20	-	-	47.09	-	-	-	-	81.76
NER	morf	73.29	76.58	83.40	77.01	65.22	84.29	86.94	59.49	74.37	81.87	66.67	90.01
	charn	**83.02**	**81.59**	**93.22**	**88.23**	**74.47**	**91.08**	88.95	**84.99**	**83.56**	**88.70**	**72.92**	94.68
	bpe1e3	77.22	79.33	89.00	85.82	71.91	89.73	89.18	81.03	81.63	85.30	70.84	92.35
	bpe1e4	76.43	79.73	89.00	85.44	65.22	89.25	88.48	70.59	80.26	86.39	64.07	92.47
	bpe1e5	-	80.65	89.36	84.02	-	88.66	**89.48**	-	81.64	86.12	68.95	93.07
	ft	73.29	79.81	88.57	86.88	58.16	89.48	89.18	58.16	81.64	83.54	68.29	92.58
	w2v	69.57	79.66	87.50	82.97	62.37	87.81	87.99	58.56	79.43	84.21	61.37	89.57
	bert	-	-	-	82.31	-	-	88.45	-	-	-	-	**95.53**

Table 3: Test accuracy for FGET and test F1 score NER for the 12 low-resource test languages. The results are obtained by training on the full WE data (except for BERT) and the full task data of the corresponding languages.

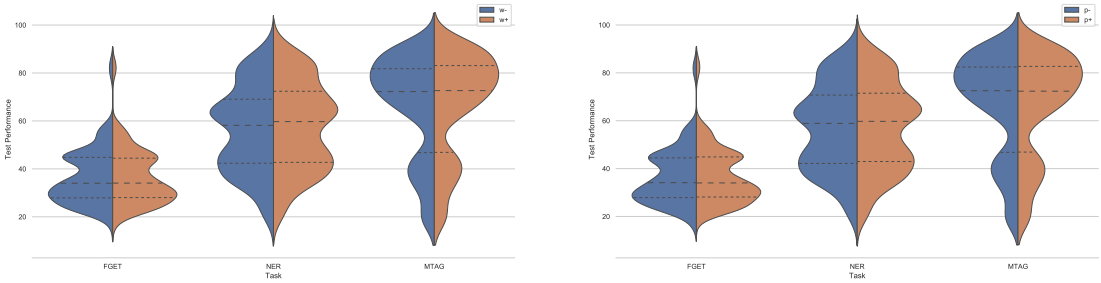


Figure 4: Comparisons of configurations with and without word token (*w-*, *w+*) and the position embedding (*p-*, *p+*). The results are obtained by collecting all data points in the data scarcity simulation for four high resource languages and the other 12 languages with both full WE data and task data.

	Agg	Fus	Int
	TE	GOT	MT
morf	87.79	76.94	92.80
charn	90.29	**85.71**	**94.39**
bpe1e3	90.01	82.07	94.16
bpe1e4	87.79	83.28	92.82
bpe1e5	87.10	-	92.51
ft	**90.85**	76.50	93.91
w2v	85.71	29.63	90.43
bert	87.45	-	-

Table 4: Test accuracy for MTAG for low-resource languages from UD where train/dev/test sets are available.

sentation architectures for truly low-resource languages. Our experiments on three diverse morphological tasks with 16 typologically diverse languages of varying degrees of data scarcity have validated that subword-level knowledge is indeed crucial for improved task performance in such low-data setups. The large amount of results reported in this work has enabled comparisons of different subword-informed methods in relation to multiple aspects such as the degree of data scarcity (both in terms of embedding training data and task-specific annotated data), the task at hand, the actual language, as well as the methods' internal design (e.g. the choice of the segmentation method). Our results have demonstrated that all these aspects must be considered in order to identify an optimal subword-informed representation architecture for a particular use case, that is, for a particular language (type), task, and data availability. However, similar paterns emerge: e.g., resorting to a segmentation method based on character n-grams seems most robust across the three tasks and across languages, although there are clear outliers. In future work, we will extend our focus to other target languages, including the ones with very limited (Adams et al., 2017) or non-existent digital footprint.

Acknowledgments

This work is supported by the ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909) and the Klaus Tschira Foundation, Heidelberg, Germany. We thank the three anonymous reviewers for their helpful suggestions.

References

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. 2017. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of EACL*, pages 937–947.

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the ACL*, 4:301–312.

Željko Agić, Barbara Plank, and Anders Søgaard. 2017. Cross-lingual tagger evaluation without test data. In *Proceedings of EACL*, pages 248–253.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of COLING*, pages 1638–1649.

Oded Avraham and Yoav Goldberg. 2017. The interplay of semantics and morphology in word embeddings. In *Proceedings of EACL*, pages 422–426.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the ACL*, 5:135–146.

Aditi Chaudhary, Chunting Zhou, Lori Levin, Graham Neubig, David R. Mortensen, and Jaime Carbonell. 2018. Adapting word embeddings to new languages with morphological and phonological subword representations. In *Proceedings of EMNLP*, pages 3285–3295.

Ryan Cotterell and Georg Heigold. 2017. Cross-lingual character-level neural morphological tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 748–759, Copenhagen, Denmark. Association for Computational Linguistics.

Ryan Cotterell, Sebastian J. Mielke, Jason Eisner, and Brian Roark. 2018. Are all languages equally hard to language-model? In *Proceedings of NAACL-HLT*.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word-embeddings. In *Proceedings of NAACL-HLT*, pages 1287–1292.

Ryan Cotterell and Hinrich Schütze. 2018. Joint semantic synthesis and morphological analysis of the derived word. *Transactions of the ACL*, 6:33–48.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*.

Nadir Durrani, Fahim Dalvi, Hassan Sajjad, Yonatan Belinkov, and Preslav Nakov. 2019. One size does not fit all: Comparing NMT representations of different granularities. In *Proceedings of NAACL-HLT*, pages 1504–1516.

Meng Fang and Trevor Cohn. 2017. Model transfer for tagging low-resource languages using a bilingual dictionary. In *Proceedings of ACL*, pages 587–593.

Philip Gage. 1994. A new algorithm for data compression. *C Users J.*, 12(2):23–38.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of ICML*, pages 1243–1252.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart, and Anna Korhonen. 2018. On the relation between linguistic typology and (limitations of) multilingual language modeling. In *Proceedings of EMNLP*, pages 316–327.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of LREC*, pages 3483–3487.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of LREC*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Yeachan Kim, Kang-Min Kim, Ji-Min Lee, and SangKeun Lee. 2018. Learning to generate word representations using subword information. In *Proceedings of COLING*, pages 2551–2561.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

András Kornai. 2013. Digital language death. *PloS One*, 8(10):e77056.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of ACL*, pages 66–75.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proceedings of EMNLP*, pages 5039–5049.

Angeliki Lazaridou, Marco Marelli, Roberto Zamparelli, and Marco Baroni. 2013. Compositional-ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of ACL*, pages 1517–1526.

Bofang Li, Aleksandr Drozd, Tao Liu, and Xiaoyong Du. 2018. Subword-level composition functions for learning word embeddings. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 38–48.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of AAAI*, pages 94–100.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of ACL*, pages 1054–1063.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of LREC*, pages 1659–1666.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of ACL*, pages 1946–1958.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proccedings of NAACL-HLT*, pages 2227–2237.

Mohammad Taher Pilehvar, Dimitri Kartsaklis, Victor Prokhorov, and Nigel Collier. 2018. Card-660: Cambridge rare word dataset - a reliable benchmark for infrequent word representation models. In *Proceedings of EMNLP*, pages 1391–1401.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword RNNs. In *Proceedings of EMNLP*, pages 102–112.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of ACL*, pages 412–418.

Edoardo Maria Ponti, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen. 2018. Modeling language variation and universals: A survey on typological linguistics for natural language processing. *arXiv preprint arXiv:1807.00914*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*, pages 1715–1725.

Peter Smit, Sami Virpioja, Stig-Arne Grnroos, and Mikko Kurimo. 2014. Morfessor 2.0: Toolkit for statistical morphological segmentation. In *Proceedings of EACL*, pages 21–24.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of ACL*, pages 2016–2027.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledge base. *Communications of the ACM*, 57:78–85.

Shijie Wu and Mark Dredze. 2019. Beto, Bentz, Becas: The surprising cross-lingual effectiveness of BERT. *CoRR*, abs/1904.09077.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of EMNLP*, pages 715–725.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2018. Subword-augmented embedding for cloze reading comprehension. In *Proceedings of COLING*, pages 1802–1814.

Jinman Zhao, Sidharth Mudgal, and Yingyu Liang. 2018. Generalizing word embeddings using bag of subwords. In *Proceedings of EMNLP*, pages 601–606.

Yi Zhu, Ivan Vulić, and Anna Korhonen. 2019. A systematic study of leveraging subword information for learning word representations. In *Proceedings of NAACL-HLT*.

Comparing Top-down and Bottom-up Neural Generative Dependency Models

Austin Matthews and **Graham Neubig**
Language Technologies Institute
Carnegie Mellon University
{austinma,gneubig}@cs.cmu.edu

Chris Dyer
DeepMind
cdyer@google.com

Abstract

Recurrent neural network grammars (RNNGs) generate sentences using phrase-structure syntax and perform very well in terms of both language modeling and parsing performance. However, since dependency annotations are much more readily available than phrase structure annotations, we propose two new generative models of projective dependency syntax, so as to explore whether generative dependency models are similarly effective. Both models use RNNs to represent the derivation history with making any explicit independence assumptions, but they differ in how they construct the trees: one builds the tree bottom up and the other top down, which profoundly changes the estimation problem faced by the learner. We evaluate the two models on three typologically different languages: English, Arabic, and Japanese. We find that both generative models improve parsing performance over a discriminative baseline, but, in contrast to RNNGs, they are significantly less effective than non-syntactic LSTM language models. Little difference between the tree construction orders is observed for either parsing or language modeling.

1 Introduction

Recurrent neural network grammars (Dyer et al., 2016, RNNGs) are syntactic language models that use predicted syntactic structures to determine the topology of the recurrent networks they use to predict subsequent words. Not only can they learn to model language better than non-syntactic language models, but the conditional distributions over parse trees given sentences produce excellent parsers (Fried et al., 2017).

In this paper, we introduce and evaluate two new dependency syntax language models which are based on a recurrent neural network (RNN) backbone (§2).[1] Dependency syntax is particularly appealing as many more languages have dependency treebanks (e.g. the Universal Dependencies Project (Nivre et al., 2017)) than have large numbers of phrase structure annotations.

Like RNNGs, our proposed models predict structure and words jointly, and the predicted syntactic structure is used to determine the structure of the neural network that is used to represent the history of actions taken by the model and to make a better estimate of the distribution over subsequent structure-building and word-generating actions. Because we use RNNs to encode the derivation history, our models do not make any explicit independence assumptions, but instead condition on the complete history of actions. The two proposed models do, however, differ in the order that they construct the trees. The first model operates top down (§2.1), starting at the root and recursively generating dependents until the last modifier has been generated. The second operates bottom up (§2.2), generating words from left to right and interleaving decisions about how they fit together to form tree fragments and finally a fully formed dependency tree.[2]

Because neither model makes explicit independence assumptions, given enough capacity, infinite data, and a perfect learner, both models would converge to the same estimate. However, in our limited, finite, and imperfect world, these two models will impose different biases on the learner: in one order, relevant conditioning information may be more local (which could mean the neural networks have an easier time learning to exploit the relevant information rather than becom-

[1] We release code for these two models, which can be found at https://github.com/armatthews/dependency-lm.

[2] In this work, we limit ourselves to models that are capable only of generating projective dependency trees.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 227–237
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

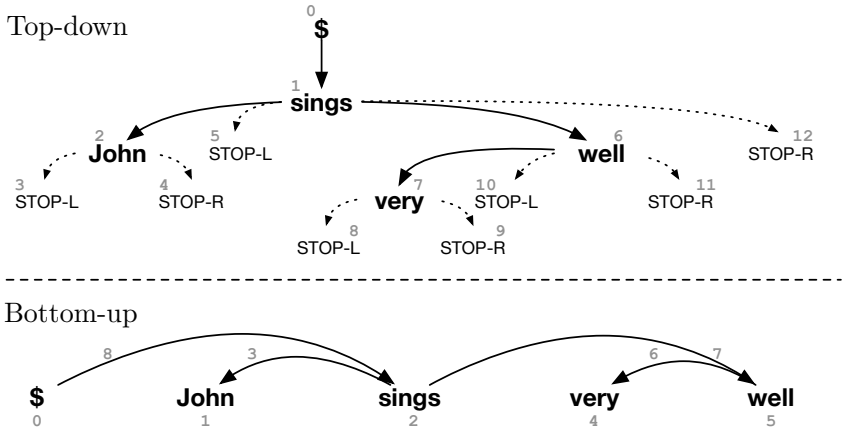


Figure 1: Generation process for the same dependency tree under the top-down and bottom-up models. As the indices of the generation events show, the top-down model generates recursively from the root, whereas the bottom-up model generates from left to right.

ing distracted by accidental correlations), while in the other it may be more distant. These differences thus imply that the two models will have different structural biases, but it is not at all clear whether one should out perform the other. We therefore explore to what extent this choice of construction order affects performance, and we evaluate the proposed models on language modeling and parsing tasks across three typologically different languages (§3).

Our findings (§4) show that, like RNNGs, generative dependency models make good parsers. Given the small scale of the Universal Dependency corpora, this result is also in line with previous work which shows that joint generative models offer very sample-efficient estimates of conditional distributions (Yogatama et al., 2017). Second, we find that both dependency models are *less effective* as language models than phrase structure RNNGs or than standard LSTM language models. This negative result is not entirely surprising. Although information about syntactic dependencies seems intuitively that it would be helpful for defining good conditioning contexts for language models, since its earliest days (Tesnière, 1959), work on dependency syntax has largely focused on discriminative models of *existing* sentences. In contrast, the phrase structure annotations found in, e.g., the Penn Treebank that were used to demonstrate improved language modeling performance with RNNGs are indebted to linguistic theories (e.g., government and binding theory, X-bar theory) which are broadly concerned with determining which sentences are grammatical and which are not—a crucial aspect of language modeling (Marcus et al., 1993). Finally, we observe only minimal differences in language modeling performance for top-down and bottom-up models. This result is surprising in light of how different the estimation problems are, but it is a clear demonstration of the ability of RNNs to learn to extract relevant features from data presented in any different but consistent orders.

2 Models

We present two models for jointly generating projective dependency trees and sentences. The processes are illustrated in Fig. 1. The first is a top-down model (§2.1), which starts by generating the root of the sentence, and then recursively generating its left and right modifiers. The second is a bottom-up model (§2.2), which generates terminals in a left to right order. In both cases, there is a deterministic mapping from well-formed sequences of generation actions into dependency trees. Following convention in parsing literature, we refer to such action sequences as oracles.

Both models both are parameterized with recursively structured neural networks that have access to the complete history of generation events. Thus, the factorization of the tree probability is justified by the chain rule. However, because of the difference in build orders, the conditional probabilities being estimated are quite different, and we thus expect these models might be more or less effective at either language modeling or (when used in conjunction with Bayes' rule) parsing.

To illustrate the different estimation problems posed by the two models, consider the first generation event in both cases. In the top-down model,

the root word (usually the main verb of the sentence) is generated first; whereas in the bottom-up model, the first probability modeled is the probability of the first word in the sentence. Also, in the top-down model a verb always generates its dependentents (which has implications for how agreement is modeled), whereas in the bottom-up model, it the left dependents (whatever their function) will be generated first, and then the verb generation will be conditional on them. Again, we emphasize that these differences potentially result in differences in the difficulty of the estimation problem (or how much capacity the model needs to represent accurate conditionals), but do not impact the expressivity or correctness of the models.

2.1 Top-Down

Our first model is a top-down model. The model begins with an empty root node.[3] Starting from the root the model recursively generates child nodes using its GEN action. When the model chooses the GEN action it then selects a new head word from its vocabulary and creates a new node descended from the most recent open constituent. Each node created in this way is pushed onto a stack of open constituents, and begins creating its left children.

To indicate that the current node (i.e. the top node in the stack) is done generating left children the model takes its STOP-L ("stop left") action, after which the current node begins generating its right children. Analogously, to indicate that the current node is done generating right children the model selects its STOP-R ("stop right") action. With this the current constituent is complete and thus popped off the stack and attached as a child of the new top-most constituent. See Figure 2 for examples of the effects of each of these three actions on a partially built tree structure and Algorithm 1 for a sketch of their implementation.

At each decision point the model conditions on the output of an LSTM over the partially completed constituents on the stack, beginning with the root and ending with the top-most constituent. The result is passed through an MLP and then a softmax that decides which action to take next (Figure 3). If the model chooses the GEN action, the hidden vector from the MLP is used to separately choose a terminal.

[3]The root node may never have left children. In this way it is though the root node has already generated its STOP-L, though this step is not explicitly modelled

Algorithm 1 Top-Down Tree Generation

```
1:  procedure EMBEDTREE(node)
2:      state = lstm_initial_state
3:      for child in node do
4:          if child is terminal then
5:              state.add(WordEmbs[child])
6:          else
7:              state.add(EMBEDTREE(child))
8:      return state
9:  procedure PICKNEXTACTION(stack)
10:     h = MLP_action(EmbedTree(stack))
11:     action ~ softmax(h)
12:     return action
13: procedure PICKWORD(stack)
14:     h = MLP_word(EmbedTree(stack))
15:     word ~ softmax(h)
16:     return word
17: procedure GENERATENODE(stack)
18:     action = PICKNEXTACTION(stack)
19:     if action == GEN then
20:         word = PICKWORD(STACK)
21:         stack.push(new Node(word))
22:     else if action == STOP-L then
23:         stack.back().add_child(STOP-L)
24:     else if action == STOP-R then
25:         stack.back().add_child(STOP-R)
26:         child_emb = stack.pop()
27:         stack.back().add_child(child_emb)
```

To embed each subtree on the stack we use another LSTM. First we feed in the head word of the constituent, followed by the embeddings of each of the constituent's children, including the special STOP-L and STOP-R symbols. We then additionally add a gated residual connection from the head word to final subtree representation to allow salient information of the head word to be captured without needing to pass through an arbitrary number of LSTM steps (Figure 4).

2.2 Bottom-Up

Our second model generates sentences bottom-up, in the same manner as a shift-reduce parser. A sentence is modeled as a series of actions (related to the arc-standard transitions used in parsing (Nivre, 2013)) that manipulate a stack of embedded tree fragments. There are three types of actions: SHIFT(x), which pushes a new terminal x onto the stack, REDUCE-L, which combines the two top elements on the stack into one single sub-

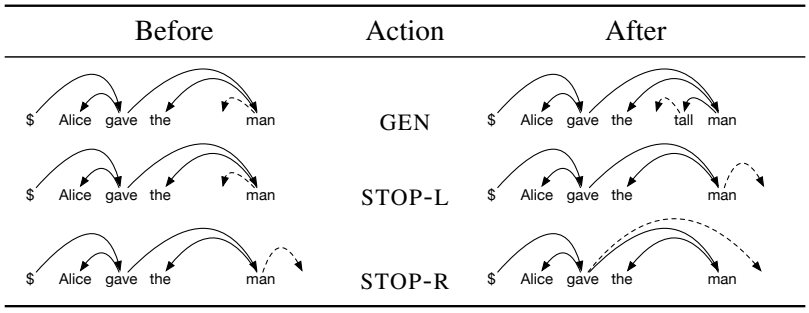


Figure 2: Examples of the three actions of our top-down model. The dotted arrow indicates where the new word will go if the GEN action is chosen next. GEN creates a new terminal node and moves the dotted arrow to point to the left of the new token. STOP-L moves the dotted arrow from the left of the current token to the right thereof. STOP-R moves the dotted arrow from the right of the current token back up to its parent node.

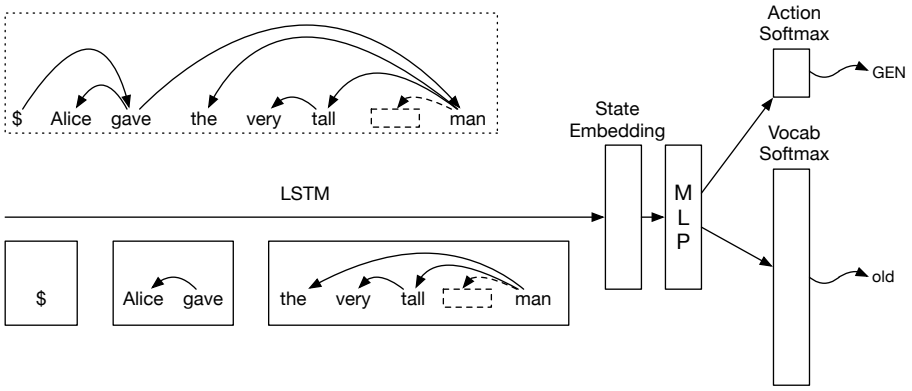


Figure 3: To encode the history of generation events in the top-down process, we use an LSTM over *subtree embeddings* (See Figure 4). The LSTM proceeds from the root of the tree down to the most recent open node. Each item in the LSTM is an embedding of a word and its already generated descendants. STOP symbols have been suppressed for clarity.

tree with the left of the two as the head (i.e. with a leftward arrow), and REDUCE-R which again combines the top two elements of the stack, this time making the right one the head. See Figure 5 for examples of how these three actions affect the stack of partially built tree structures during the parsing of an example sentence.

At each time step the model conditions on the state of the stack using an LSTM running over entries from oldest to newest. The resulting vector $\boldsymbol{h}$ is then passed through an MLP, and then a softmax over the three possible action types. If the SHIFT action is taken, the vector $\boldsymbol{h}$ is re-used and passed through a separate MLP and softmax over the vocabulary to choose an individual word to generate. If one of the two REDUCE actions is chosen, the top two elements from the stack are popped, concatenated (with the head-to-be first, followed by the child), and passed through an MLP. The result is a vector representing a new subtree that is then pushed onto the stack. Kuncoro et al. (2017) showed that this type of stack-based representation alone is sufficient for language modeling and parsing, and indeed that more involved models actually damage model performance. See Figure 6 for an example of how this bottom-up model chooses an action.

2.3 Marginalization

Traditionally a language model takes a sentence $\boldsymbol{x}$ and assigns it a probability $p(\boldsymbol{x})$. Since our syntax-based language models jointly predicts the probability $p(\boldsymbol{x}, \boldsymbol{y})$ of a sequence of terminals $\boldsymbol{x}$ and a tree $\boldsymbol{y}$, we must marginalize over trees to get the total probability assigned to a sentence $\boldsymbol{x}$, $p(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{T}(\boldsymbol{x})} p(\boldsymbol{x}, \boldsymbol{y})$, where $\mathcal{T}(\boldsymbol{x})$ represents the set of all possible dependency trees over a sentence $\boldsymbol{x}$. Unfortunately the size of $\mathcal{T}(\boldsymbol{x})$ grows exponentially in the length of $\boldsymbol{x}$, making explicit marginalization infeasible.

Instead we use importance sampling to approximate the marginal (Dyer et al., 2016). We use the

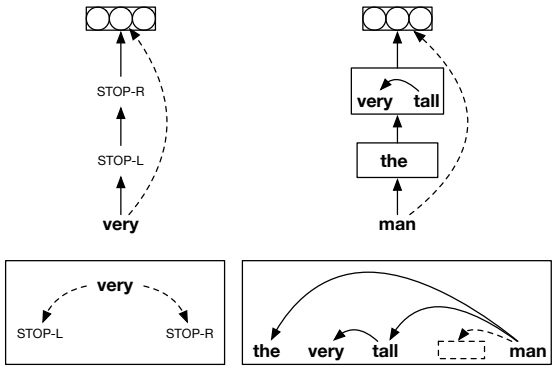


Figure 4: Examples of embedding two subtrees in the top-down model. A subtree is embedded using an LSTM over its child subtrees (solid lines) with a gated residual connection from the root word to the final embedding (dotted lines).

parser of Dyer et al. (2015), a discriminative neural stack-LSTM-based bottom-up parser, as our proposal distribution $q(\boldsymbol{x}, \boldsymbol{y})$ and compute the approximate marginal using $N = 1000$ samples per sentence: $p(\boldsymbol{x}) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{p(\boldsymbol{x},\boldsymbol{y})}{q(\boldsymbol{x},\boldsymbol{y})}$.

2.4 Parsing Evaluation through Reranking

In order to evaluate our model as a parser we would ideally like to efficiently find the MAP parse tree given an input sentence. Unfortunately, due to the unbounded dependencies across the sequences of actions used by our models this inference is infeasible. As such, we instead rerank a list of 1000 samples produced by the baseline discriminative parser, a combination process that has been shown to improve performance by combining the different knowledge learned by the discriminative and generative models (Fried et al., 2017).

For each hypothesis parse in the sample list we query the discriminative parser, our top-down model, and our bottom-up model to obtain a score for the parse from each. We combine these scores using weights learned to optimize performance on the development set (Och, 2003).

3 Experimental Setup

Our primary goal is to discover whether dependency-based generative neural models are able to improve the performance of their discriminative brethren, as measured on parsing and language modeling tasks. We also seek to determine the effect construction order and the biases implicit therein has on performance on these two tasks. To this end, we test a baseline discriminative parser, our two models, and all combinations of these three models on a parsing task in several languages, and we test a baseline and our two models' performance on a language modeling task on the same set of languages.

3.1 Data Sets

We use the Universal Dependency corpora (Nivre et al., 2017) for three languages with very different structures: English, Japanese, and Arabic, as provided for the 2017 CoNLL shared task on universal dependency parsing. In all languages we convert all singleton terminal symbols to a special UNK token. See Table 1 for details regarding the size of these data sets.

For language modeling we evaluate using the gold sentence segmentations, word tokenizations, and part of speech tags given in the data. For parsing, we evaluate in two scenarios. In the first, we train and test on the same gold-standard data using in our language modeling experiments. In the second, we again train on gold data, but use UDPipe (Straka and Straková, 2017) to segment, tokenize, and POS tag the dev and test sets starting from raw text, following the default scenario and most participants in the CoNLL 2017 shared task.

3.2 Baseline Models

On the language modeling task we compare against a standard LSTM-based language model baseline (Mikolov et al., 2010), using 1024-dimensional 2-layer LSTM cells, and optimized using Adam (Kingma and Ba, 2014).

For the parsing task we compare against the discriminative parser of Dyer et al. (2015), a bottom-up transition-based parser that uses stack-LSTMs, as well as the overall top system (Dozat et al., 2017) from the 2017 CoNLL shared task on multilingual dependency parsing (Zeman et al., 2017). That work uses a discriminative graph-based parser that uses a biaffine scoring function to score each potential arc. Moreover, it uses character-level representations to deal with morphology and a PoS tagger more sophisticated than UDPipe – two major changes from the shared task's default pipeline. These two differences afford them a substantial advantage over our approach which only modifies the parsing step of the pipeline.

Finally, we show the results of an oracle system looking at the 1000-best lists used for our reranking experiments. Note that since this oracle system

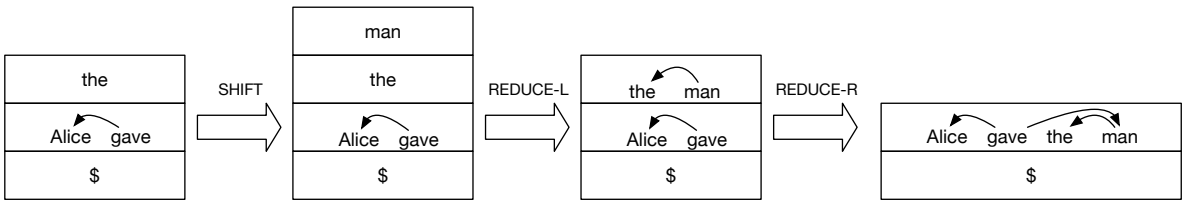


Figure 5: Examples of the three actions of our bottom-up model and their effects on the internal stack. SHIFT adds a new terminal to the top of the stack. REDUCE-L combines the top two elements of the stack with a left arc from the head of the top-most element to the head of the second element. REDUCE-R combines the top two elements with a right arc from the head of the second element to the head of the top-most element.

		Train		Dev		Test		Vocab
Language	Words	Sents	Words	Sents	Words	Sents	Singletons	Non-S'tons
English	204585	12543	25148	2002	25096	2077	9799	9873
Japanese	161900	7164	11556	511	12615	557	13091	9222
Arabic	223881	6075	30239	909	28264	680	9907	13242

Table 1: Statistics of the universal dependency data sets used in this paper. Size of the train, dev, and test sets are given in tokens. Vocabulary information is number of types.

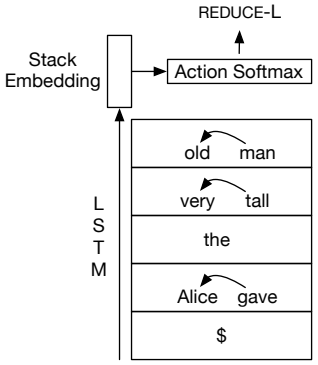


Figure 6: Our bottom-up model emulates a shift-reduce parser and maintains an explicit stack. At each timestep, we use the output of an LSTM over the stack to choose the next action, which is then executed to produce a new stack state.

is constrained to using only this list of samples it is not able to achieve 100% parsing accuracy.

3.3 Hyperparameters

All models use two-layer 1024-unit LSTMs and 1024-dimensional word/action embeddings. All other MLPs have a single hidden layer, again with 1024 hidden units. We implement all models using DyNet (Neubig et al., 2017), and train using Adam (Kingma and Ba, 2014) with a learning rate of 0.001, dropout with $p = 0.5$, and minibatches of 32 sentences. We evaluate the model on a held out dev set after 150 updates, and save the model to disk whenever the score is a new best. All other settings use DyNet defaults.

4 Results

Parsing results Results on the parsing task can be found in Table 2. We observe that in English with the gold-standard preprocessing our models perform particularly well, showing an improvement of 1.16% UAS F1 for the top-down and 0.82% UAS F1 for the bottom-up model when individually combined with our discriminative parser. Combining all three models together gives a total of 1.46% absolute improvement over the baseline, indicating that the models capture knowledge lacking in the baseline model, and knowledge that is complementary to each other.

The story is similar in Japanese and Arabic, though the gains are smaller in Japanese. We hypothesize that this is due to the fact that parsing Japanese is relatively easy because of its strict head-final and left-branching nature, and thus our baseline is already a remarkably strong parser. This hypothesis is backed up by the fact that the baseline parser alone is only 3-4% UAS away from the oracle by itself, compared to about 10% away on English and Arabic. Thus our relative improvement, measured in terms of the percentage of possible improvement achieved, is quite consistent across the three languages, at roughly 13%.

Results on the test set using UDPipe's noisy preprocessing also saw encouraging results from

		English				Japanese				Arabic			
		Gold → Gold		Gold → UDPipe		Gold → Gold		Gold → UDPipe		Gold → Gold		Gold → UDPipe	
Model	Reranked?	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
CoNLL Baseline	✗	-	-	-	79.24	-	-	-	74.40	-	-	-	70.14
Dozat et al. (2017)	✗	-	-	-	84.74	-	-	-	75.42	-	-	-	76.59
Disc (Greedy)	✗	87.00	85.92	78.85	78.12	96.16	95.20	76.67	75.70	82.14	82.39	69.74	70.34
Disc (Reranked)	✓	87.48	86.30	78.99	78.23	96.04	95.17	76.66	75.58	81.70	81.34	69.20	68.79
Top-Down	✓	82.94	82.48	76.73	76.88	92.99	92.51	74.84	74.18	80.99	80.85	69.78	69.64
Bottom-Up	✓	83.11	82.70	76.79	77.13	94.56	93.25	75.85	74.18	80.61	80.70	69.30	69.24
Disc + TD	✓	88.47	87.46	80.46	79.56	96.07	95.43	76.59	74.62	82.87	82.37	70.35	70.25
Disc + BU	✓	88.29	87.12	80.09	79.33	96.17	95.54	76.82	75.92	82.48	82.18	70.18	69.99
TD + BU	✓	84.93	84.56	78.71	78.72	94.87	94.03	76.15	75.30	81.84	81.56	70.52	70.03
Disc + TD + BU	✓	**88.74**	**87.76**	**80.49**	**80.22**	**96.18**	**95.58**	**76.86**	**75.98**	**83.06**	**82.58**	**70.85**	**70.40**
Oracle	✓	97.68	97.27	91.07	90.24	99.39	99.25	79.67	80.34	91.20	89.06	77.75	76.17

Table 2: Results of parsing using our baseline discriminative parser, our two generative models, combinations thereof, and two contrastive systems from the CoNLL 2017 shared task. Scores in bold are the highest of our models. Note that Dozat et al. (2017) use substantially different preprocessing. See §3.2 for details.

		$p(\boldsymbol{x}, \boldsymbol{y})$		$p(\boldsymbol{x})$	
Lang.	Model	Dev	Test	Dev	Test
EN	RNNLM	-	-	**5.24**	**5.18**
	Top-Down	5.80	5.72	5.73	5.66
	Bottom-Up	5.63	5.56	5.53	5.47
JA	RNNLM	-	-	**4.41**	**4.58**
	Top-Down	4.82	5.00	4.73	4.93
	Bottom-Up	4.83	5.03	4.75	4.95
AR	RNNLM	-	-	**5.42**	**4.34**
	Top-Down	6.08	6.23	5.98	4.79
	Bottom-Up	6.11	6.21	5.94	4.75

Table 3: Language modeling cross entropy of our model and an RNNLM baseline. Lower is better. All scores are expressed in nats.

the three-model ensemble gaining 1.99%, 0.40%, and 1.61% on English, Japanese, and Arabic respectively, solidly outperforming the 2017 CoNLL shared task baselines across the board, and beating Dozat et al. (2017), the overall shared task winner's, submission on Japanese.

Of particular note is that on both the gold and non-gold data, and across all three languages, the performance of the top-down and bottom-up models is quite similar; neither model consistently outperforms the other. In Japanese we do find the bottom-up parser beats the top-down one when used alone, but when combined with the discriminative model the lead evaporates, and in both of the other languages there is no clear trend.

These results are consistent with Fried et al. (2017) that has shown that generative models are particularly good at improving discriminative models through reranking, as they have an effect similar to ensembling dissimilar models.

Language modeling results We find that despite successes on parsing, our dependency models are not empirically suitable for language modeling. Table 3 shows the performance of our models on the language modeling task. Across all three languages, both of our models underperform a baseline RNNLM by a consistent margin of about 0.5 nats per word.

Again we note that the two models perform remarkably similarly, despite their completely different construction orders, and thus the completely different sets of information they condition on at each time step. Again neither model is a clear overall victor, and in each individual language the models are extremely close in performance.

5 Analysis

One of our most intriguing findings is that our two proposed models perform remarkably similarly in spite of their differing construction orders. One would naturally assume that the differing orders, as well as the wildly different history information available at each decision point, would lead to performance differences. We seek to hone in on *why* the two models' performances are so similar.

Unfortunately the fact that the models use different conditioning contexts makes direct comparison of sub-sentential scores impossible. The top-down model, which generates the verb before its subject noun, may have large entropy when choosing the verb, but an easier time choosing the subject since it can condition on the verb limiting its choices to appropriate semantic classes, person,

	Structure	Terminals
Top-Down	4.97	67.9
Bottom-Up	7.42	63.3

Table 4: Our models' average negative log likelihoods on the English dev set broken down into structure and terminal components

number, et cetera. The bottom-up model, on the other hand, will generate the subject noun from the entire list of possible nouns first, and then will focus its probability on relevant and agreeing verb forms when generating the verb.

To this end we plot the scores (i.e. the negative log probabilities) the models assign to each gold tree in the English dev set. The raw scores between the top-down and bottom-up models are highly correlated (Pearson's $r = 0.995$), largely due to the fact that longer sentences naturally have lower probabilities than shorter sentences. As such, we examine length-normalized scores, dividing each sentence's score by its length. The results are still largely correlated ($r = 0.88$), with a few outliers, all of which are very short (<3 tokens) sentences.

We hypothesize that much of this correlation stems from the fact that for a given sentence both models must generate the same sequence of terminal symbols. Some sentences will have rare sequences of terminals while others have more common words, leading to an obvious, but perhaps uninformative, correlation. To examine this possibility we factor our models' scores into a *terminal* component and a *structure* component so that the overall negative log likelihood of a sentence is decomposed as $\text{NLL} = \text{NLL}_{\text{terminals}} + \text{NLL}_{\text{structure}}$. We then examine the correlation between the two models' scores' terminal components and their structure components separately. We find that the terminal components are still strongly correlated ($r = 0.91$), while the structure components are largely uncorrelated ($r = 0.09$), hinting that information the two models learned about the correct structure of English sentences differs. See Appendix Figure 7 for a visual representation of these data. Overall the top-down model also assigns much higher probabilities to correct structures, but lower probabilities to the correct terminal sequences (Table 4).

6 Related Work

Most work on discriminative dependency parsing follows the bottom-up paradigm (Nivre, 2003; Nivre et al., 2007; Dyer et al., 2015; Kiperwasser and Goldberg, 2016), but top-down models have also shown some promise (Zhang et al., 2015).

Generative dependency models go back to Hays (1964), but most existing such models (Buys and Blunsom, 2015; Jiang et al., 2016) have relied on independence assumptions whether used for parsing, unsupervised dependency induction, or language modeling. Buys and Blunsom (2015) also describe a generative bottom-up neural parser, but use hand-crafted input features and limit the model to third-order features. Titov and Henderson (2010) explore a generative parsing model with no independence assumptions based on sigmoid belief networks (Neal, 1992) instead of RNNs.

The CoNLL 2017 shared task saw many different models succeed at parsing Universal Dependencies. Most of the top contenders, including the best scoring systems on the languages discussed in this work, use discriminative models.

Kanayama et al. (2017) had tremendous success on Japanese using a wildly different approach. They train a model to identify likely syntactic heads, then assume that all other words simply attach in a left-branching structure, which works due to the strictly head-final nature of Japanese.

Dozat et al. (2017) train a discriminative neural parser which uses a BiLSTM to generate hidden representations of each word (Kiperwasser and Goldberg, 2016). These representations are used to score arcs, which are greedily added to the tree.

Björkelund et al. (2017) perform best on Arabic, using an ensemble of many different types of bottom-up discriminative parsers. They have each of twelve parsers score potential arcs, learn a weighting function to combine them, and use the Chu-Liu-Edmonds algorithm (Chu, 1965; Edmonds, 1967) to output final parses.

All three of these discriminative models are very effective for analysis of a sentence, none of them are able to be converted into a similar generative model. At best, the biaffine model of Dozat et al. (2017) could generate a bag of dependencies without order information, which makes it impractical as the basis for a generative model.

There has been past work on building recurrent neural models that condition on the buffer to make parsing decisions in a shift-reduce parser. Hender-

son (2004) was among the first to introduce such a model. They introduce both a generative and discriminative model based on Simple Synchrony Networks (Lane and Henderson, 1998), and use a pre-cursor to attention mechanisms to choose which previous states are most relevant at the current timestep. More recently Dyer et al. (2015) created a similar model based on stack LSTMs.

There has also been past work on language modelling with generation orders other than the typical left-to-right. Ford et al. (2018) examine a variety of possibilities, but stop short of syntax-aware oderings. Buys and Blunsom (2018) investigate neural language models with latent dependency structure, also concluding that while dependencies perform well on parsing they underperform for language modelling.

7 Conclusion

In this paper we test our hypothesis that dependency structures can improve performance on language modeling and machine translation tasks, in the same way that constituency parsers have been shown to help. We conclude that generative dependency models do indeed make very good parsing models, and, as has been observed in phrase structure parsing, combining a generative dependency parser with a traditional discriminative one does indeed improve parsing performance. We however also find using dependency models information to structure the intermediate representations in language modeling does not easily lead to better outcomes.

This pattern of results suggests that while dependencies may be a useful tool for text analysis, but are less suited to characterizing a generation process of sentences than phrase structure grammars area. Finally, we find that the choice of top-down or bottom-up construction order affects performance minimally on both the parsing and language modeling tasks despite the large differences in the local conditioning contexts of each action choice.

Acknolwedgements

This work is sponsored in part by Defense Advanced Research Projects Agency Information Innovation Office (I2O). Program: Low Resource Languages for Emergent Incidents (LORELEI). Issued by DARPA/I2O under Contract No. HR0011-15-C0114. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Anders Björkelund, Agnieszka Falenska, Xiang Yu, and Jonas Kuhn. 2017. Ims at the conll 2017 ud shared task: Crfs and perceptrons meet neural networks. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 40–51.

Jan Buys and Phil Blunsom. 2015. Generative incremental dependency parsing with neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 863–869.

Jan Buys and Phil Blunsom. 2018. Neural syntactic generative models with exact marginalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 942–952. Association for Computational Linguistics.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

Timothy Dozat, Peng Qi, and Christopher D Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. ACL*.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. 2016. Recurrent neural network grammars. In *Proceedings of NAACL-HLT*, pages 199–209.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71:233–240.

Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George Dahl. 2018. The importance of generation order in language modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2942–2946.

Daniel Fried, Mitchell Stern, and Dan Klein. 2017. Improving neural parsing by disentangling model combination and reranking effects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 161–166.

David G Hays. 1964. Dependency theory: A formalism and some observations. *Language*, 40(4):511–525.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 95. Association for Computational Linguistics.

Yong Jiang, Wenjuan Han, and Kewei Tu. 2016. Unsupervised neural dependency parsing. In *Proc. EMNLP*.

Hiroshi Kanayama, Masayasu Muraoka, and Katsumasa Yoshikawa. 2017. A semi-universal pipelined approach to the conll 2017 ud shared task. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 265–273.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2017. What do recurrent neural network grammars learn about syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1249–1258.

Peter CR Lane and James B Henderson. 1998. Simple synchrony networks: Learning to parse natural language with temporal synchrony variable binding. In *ICANN 98*, pages 615–620. Springer.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Radford M Neal. 1992. Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*. Citeseer.

Joakim Nivre. 2013. Transition-based parsing.

Joakim Nivre, Lars Ahrenberg Željko Agic, et al. 2017. Universal dependencies 2.0. lindat/clarin digital library at the institute of formal and applied linguistics, charles university, prague.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Lucien Tesnière. 1959. Eléments de linguistique structurale. *Paris, Klincksieck*, 2.

Ivan Titov and James Henderson. 2010. A latent variable model for generative dependency parsing. In *Trends in Parsing Technology*, pages 35–55. Springer.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajic, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, et al. 2017. Conll 2017 shared task: multilingual parsing from raw text to universal dependencies. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–19.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2015. Top-down tree long short-term memory networks. In *Proc. NAACL*.

Graphs

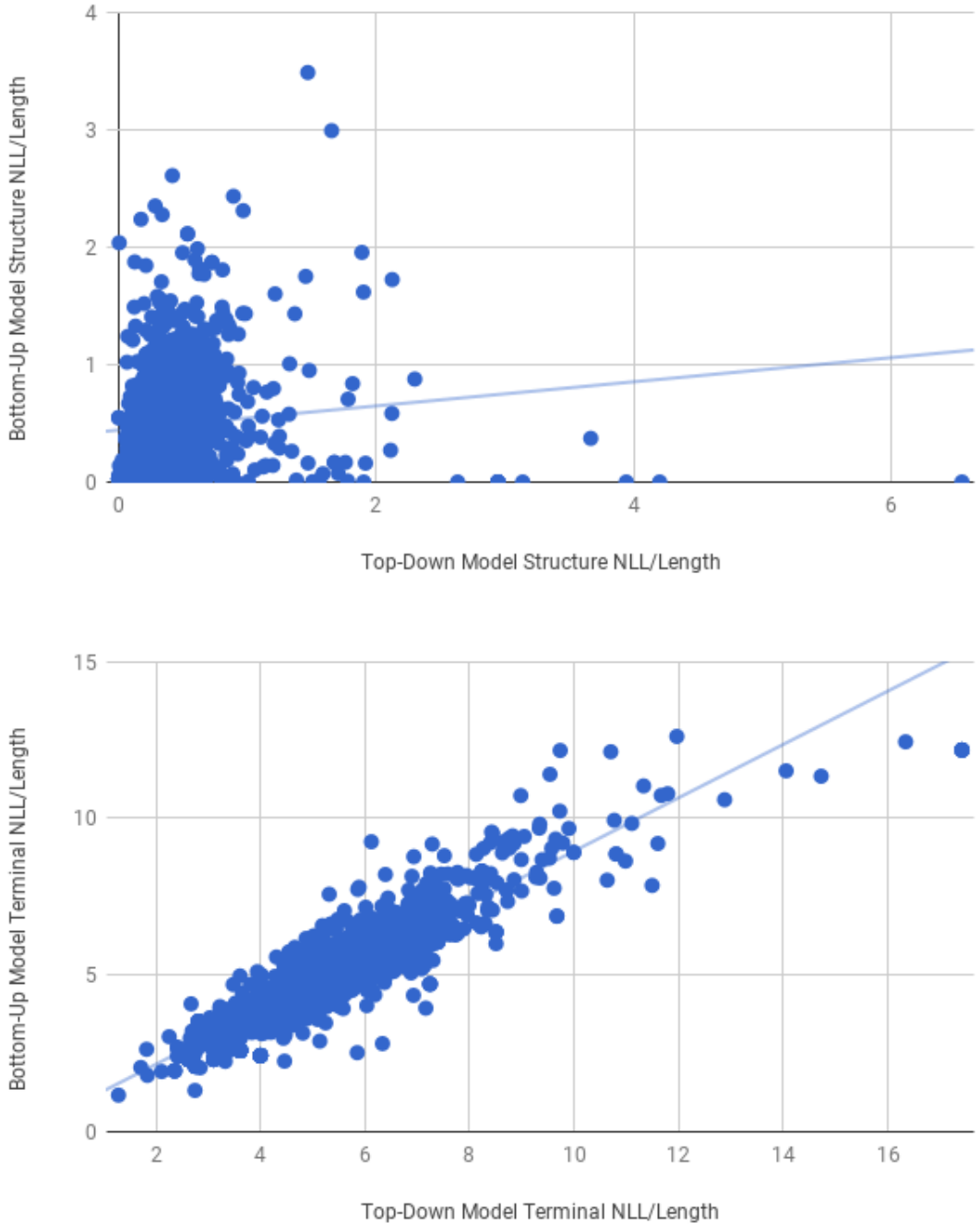


Figure 7: Analysis of the structure (top) and terminal (bottom) scores of our two models' performance on the English development set. We find that the structure scores are not correlated, while the terminal scores of the two models are highly correlated.

Representation Learning and Dynamic Programming for Arc-Hybrid Parsing

Joseph Le Roux **Antoine Rozenknop** **Mathieu Lacroix**
Laboratoire d'Informatique de Paris Nord,
Université Paris 13 – SPC, CNRS UMR 7030,
F-93430, Villetaneuse, France
{leroux,rozenknop,lacroix}@lipn.fr

Abstract

We present a new method for transition-based parsing where a solution is a pair made of a dependency tree and a derivation graph describing the construction of the former. From this representation we are able to derive an efficient parsing algorithm and design a neural network that learns vertex representations and arc scores. Experimentally, although we only train via local classifiers, our approach improves over previous arc-hybrid systems and reach state-of-the-art parsing accuracy.

1 Introduction

While transition-based dependency parsing is usually implemented as a beam-search procedure, *e.g.* (Kiperwasser and Goldberg, 2016), some recent work such as (Shi et al., 2017) showed that global inference can be performed efficiently with dynamic programming. To this end, the stack representing pending subparses and the buffer representing the unconsumed input must both be abstracted into equivalence classes, while remaining rich enough to help with accurate predictions.

In this paper we first explicitly consider that a solution in transition-based parsing is represented as a pair made of a derivation graph and a derived dependency tree allowing the scoring function to be expressed naturally as a sum over these 2 structures. While we restrict our presentation to arc-hybrid systems, our method can be applied quite directly to other transition rule systems.

Secondly we show that this representation leads to an exact $O(n^4)$ parsing algorithm using dynamic programming. This algorithm can be seen as an extension of the minimal feature set arc-hybrid parsing algorithm presented in (Shi et al., 2017) where the contribution of the dependency arcs can be explicitly added to the scoring function as in the Eisner parsing algorithm (Eisner, 1996).

We then propose an alternative approach to global inference where derivation steps are represented as dense vectors based on the number and type of steps in a derivation. With this abstraction we design a neural architecture based on non-local networks (Wang et al., 2017) related to self-attention mechanism (Vaswani et al., 2017; Gu et al., 2018) to learn these representations while maintaining the possibility for exact decoding.

Our contribution can be summarized as follows: *(i)* a representation of arc-hybrid parsing as maximum subgraphs selection where a solution contains dependencies and derivation information; *(ii)* a polynomial dynamic programming algorithm to solve this problem exactly; *(iii)* a neural architecture able to learn representations for the subgraph vertices and compute arc scores without explicit stack and buffer representations.

These contributions are validated empirically by experimental results on the Penn Treebank where our system reaches state-of-the-art accuracy (94.8% UAS) for arc-hybrid parsing with networks of comparable size.

We first review arc-hybrid dependency parsing (§2) then present a deductive scheme to solve it (§3). The neural architecture is presented in §4 and experiments reported in §5. Finally we discuss some related work in §6.

2 Arc-Hybrid Dependency Parsing

2.1 Arc-Hybrid Derivations

Intuitively, the arc-hybrid parsing strategy (Gómez-Rodríguez et al., 2008; Kuhlmann et al., 2011) builds dependency parses incrementally by reading the sentence from left to right. The pending words are words which have been given all their left modifiers but may have not been given all their right dependents yet. The pending words are stored in the stack which

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 238–248
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

initially contains a dummy root word. The words which have not been read yet are stored in order in the buffer. The first word in the buffer is called the frontier word.

The algorithm proceeds by reducing the most recent pending word (the top of the stack) which means assigning it a governor. This governor is either the current frontier word, creating a left arc, or the previously most recent pending word (below in the stack), creating a right arc. Once given a governor, the most recent pending word is popped out. When the frontier word has been given all his left dependents, it is shifted which means it is pushed in the stack, becomes the most recent pending word and the next word in the sentence becomes the frontier word.

More formally, the arc-hybrid parsing algorithm is defined using configurations. Following standard definition, a configuration is a triplet $[\sigma, \beta, A]$ where σ is a stack of indexes of words which have not been given a governor yet, β a list (buffer) of indexes of words still to be read[1], and A the set of dependency arcs that have been constructed so far.

Given sentence $w = w_1, ..., w_n$ and dummy root token w_0, there are one initial configuration $c_1 = [0, 1..., \emptyset]$ and many goal configurations of the form $[0, \emptyset, A]$. There exist 3 transition rules to pass from one configuration to another[2]:

shift $[\sigma, b|\beta, A] \rightarrow^S [\sigma|b, \beta, A]$

left $[\sigma|d, h|\beta, A] \rightarrow^L [\sigma, h|\beta, A \cup \{(h, d)\}]$

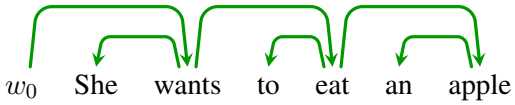
right $[\sigma|h|d, \beta, A] \rightarrow^R [\sigma|h, \beta, A \cup \{(h, d)\}]$

A derivation for w is a sequence $\gamma = c_1, t_1, c_2, t_2, \ldots, t_{2n}, c_{2n+1}$ of $2n$ transitions from c_1 leading to a goal configuration c_{2n+1}.

The set of derivations for a sentence w is noted D_w. It can be shown that derivations all generate projective dependency trees and that, for a sentence with n words, they each contain n shift operations and n left or right reductions[3].

Shifts and reductions can be seen as forming a well-parenthesized expression. Each word is associated with a kind of parenthesis and shifting and reducing this word correspond to opening and closing parentheses of this kind.

For instance, in *She wants to eat an apple*, the derivation $(_{\text{She}})^L(_{\text{wants}}(_{\text{to}})^L(_{\text{eat}}(_{\text{an}})^L(_{\text{apple}})^R)^R)^R$, where shifting a word is represented by a subscripted opening parenthesis and reductions are closing parentheses typed either L or R, will generate the following dependency tree:



Arc-hybrid parsing amounts to finding the highest-scoring derivation for a sentence:

$$\hat{\gamma} = \arg\max_{\gamma \in D_w} s(\gamma)$$

If we assume s decomposes over transition scores $s_\ell(c_\ell, t_\ell)$, we retrieve the well-studied cumulative sum of its transition scores.

$$s(\gamma) = \sum_{1 \leq \ell \leq 2n} s_\ell(c_\ell, t_\ell) \quad (1)$$

2.2 General Formulation for Dynamic Programming

We present a dynamic programming (DP) algorithm for arc-hybrid parsing with cumulative transition scores as in Eq. 1. This algorithm cannot be used as such since states references complete stack contents, of which there is an exponential number, leading to an intractable complexity. To use this algorithm in practice we would need to resort to beam search in order to approximate solutions, see for instance (Dyer et al., 2015).

However, this constitutes a general framework from which efficient algorithms can be derived by considering various equivalence classes over states and independence assumptions in the scoring function. For instance we can retrieve the *Minimal Feature Set* algorithm of Shi et al. (2017), or the new algorithm presented in Section 3.

In our algorithm, an item ${}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{j} \rangle^B$ represents the following set of subsequences of a derivation:

$${}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{j} \rangle^B : [\sigma, i..., A] \xrightarrow{*} [\sigma|i, j..., B]$$

i.e. subsequences which start with *shifting* w_i and end with w_i on top of the stack and w_j on top of the buffer[4]. As a special case, we will note ${}^{\emptyset,\emptyset}\langle \mathbf{0}, \boldsymbol{j} \rangle^B$ the set of subderivations starting from the initial configuration $[0, 1..., \emptyset]$ and leading to $[0, j..., B]$.

[1] A buffer containing $(i, ..., n)$ will be denoted by "i...".

[2] Shift and left transitions require a non-empty buffer and the stack has to be of length at least 2 for left and right transitions since w_0 cannot be the governor of a left arc.

[3] Note that derivation is not unique for a dependency tree.

[4] such a subderivation is only possible if $i < j \leq n+1$ and $A \subset B$.

Goal: Goal items have the form: ${}^{\emptyset,\emptyset}\langle \boldsymbol{0}, \boldsymbol{n{+}1}\rangle^{A} : [\emptyset, 0..., \emptyset] \xrightarrow{*} [0, \emptyset, A]$.

Axioms: The algorithm starts with the set of items corresponding to possible shifts[5]: ${}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{i{+}1}\rangle^{A} : [\sigma, i..., A] \rightarrow [\sigma|i, i{+}1..., A]$. The first axiom[6] ${}^{\emptyset,\emptyset}\langle \boldsymbol{0}, \boldsymbol{1}\rangle^{\emptyset}$ pictures a dummy sub-derivation that would put w_0 on top of the stack and lead to the initial configuration $[0, 1..., \emptyset]$.

DP Steps: A DP step consists in building an item ${}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{j}\rangle^{B}$ by composing ${}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{k}\rangle^{C}$, ${}^{\sigma|i,C}\langle \boldsymbol{k}, \boldsymbol{j}\rangle^{D}$, and a *reduce* operation on word w_k:

$$\begin{array}{rrcl} {}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{k}\rangle^{C} : & [\sigma, i..., A] & \xrightarrow{*} & [\sigma|i, k..., C] \\ {}^{\sigma|i,C}\langle \boldsymbol{k}, \boldsymbol{j}\rangle^{D} : & [\sigma|i, k..., C] & \xrightarrow{*} & [\sigma|i|k, j..., D] \\ \textit{reduce} : & [\sigma|i|k, j..., D] & \rightarrow & [\sigma|i, j..., B] \end{array}$$

We thus have a *right reduction* rule:

$$\frac{{}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{k}\rangle^{C} \quad {}^{\sigma|i,C}\langle \boldsymbol{k}, \boldsymbol{j}\rangle^{D}}{{}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{j}\rangle^{D\cup\{(i,k)\}}} \quad [(i \rightarrow k)]$$

and a *left reduction* rule:

$$\frac{{}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{k}\rangle^{C} \quad {}^{\sigma|i,C}\langle \boldsymbol{k}, \boldsymbol{j}\rangle^{D}}{{}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{j}\rangle^{D\cup\{(k,j)\}}} \quad [(k \leftarrow j)]$$

Scoring Items: We trivially score an axiom ${}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{i{+}1}\rangle^{A}$ with the score of a *shift* transition occurring in configuration $[\sigma, i..., A]$.

We compute the score of a DP step as the sum of the scores of the combined items and the score of *reducing* w_k from configuration $[\sigma|i|k, j..., D]$. When several DP steps produce the same item, it is assigned the highest score.

2.3 Minimal Feature Set Algorithm

For this algorithm (Shi et al., 2017), the score of a transition t_ℓ does not depend on the whole configuration but only on the index on top of the stack and the first index of the buffer. In other words, local scores s_ℓ only depend on word indexes (i,j,k). This assumption is crude but it allows for quite large items equivalence classes. We can retrieve this algorithm from the one above by removing unnecessary information in the items. Items of the form ${}^{\sigma,A}\langle \boldsymbol{i}, \boldsymbol{j}\rangle^{B}$ will simply reduce to $\langle \boldsymbol{i}, \boldsymbol{j}\rangle$, there will be $O(n^2)$ such items, and the DP complexity will be $O(n^3)$. More concretely, we have the following schemata:

Goal: $\langle \boldsymbol{0}, \boldsymbol{n{+}1}\rangle$.

Axioms: $\langle \boldsymbol{i}, \boldsymbol{i{+}1}\rangle$.

DP Steps:

$$\frac{\langle \boldsymbol{i}, \boldsymbol{k}\rangle \quad \langle \boldsymbol{k}, \boldsymbol{j}\rangle}{\langle \boldsymbol{i}, \boldsymbol{j}\rangle} \quad [(i \rightarrow k)]$$

and

$$\frac{\langle \boldsymbol{i}, \boldsymbol{k}\rangle \quad \langle \boldsymbol{k}, \boldsymbol{j}\rangle}{\langle \boldsymbol{i}, \boldsymbol{j}\rangle} \quad [(k \leftarrow j)]$$

One of the issues with this parsing scheme is the difficulty to interpret item scores consistently. In the case of a left reduction (producing $k \leftarrow j$ above) the score of $\langle \boldsymbol{k}, \boldsymbol{j}\rangle$ can be interpreted as the score of word j being the governor of word k and the score of $\langle \boldsymbol{i}, \boldsymbol{k}\rangle$ as the score of shifting word k in the context of i being the most recent pending word.

On the contrary, in the case of a right reduction (producing $i \rightarrow k$ above) if the score of $\langle \boldsymbol{i}, \boldsymbol{k}\rangle$ may well be interpreted analogously as the score of having i as a governor of k, we cannot interpret $\langle \boldsymbol{k}, \boldsymbol{j}\rangle$ as the score of shifting k, and it may be difficult to interpret this item score in terms of a transition operation.

3 Parsing with Derivations and Dependencies

3.1 A New Score for Derivations

We depart form previous work and make the dependency arc contribution explicit in the score function:

$$\hat{\gamma}, \hat{\tau} = \underset{\gamma,\tau \in S_w}{\arg\max}\, s(\gamma) + s(\tau) \qquad (2)$$

where S_w is the set of pairs (γ, τ) with $\gamma \in D_w$ and τ the dependency tree corresponding to γ. We use an arc-factored model for τ from now on and discuss scores for γ.

We define an equivalence class (i, q) containing all configurations c_ℓ such that the first index i_ℓ of β_ℓ equals i and the size of stack $|\sigma_\ell|$ equals q. Thus, we rewrite cumulative transition scores as:

$$s^{(u)}(\gamma) = \sum_{1 \leq \ell \leq 2n} s_u(i_\ell, |\sigma_\ell|, t_\ell). \qquad (3)$$

We also consider a score function based on the nestedness property of shift/reduce derivations. A score is given to each pair consisting of a shift and its corresponding (left or right) reduction. In other

[5] There is one axiom for each possible configuration $[\sigma, i..., A]$ with $0 \leq i < n$, σ a valid stack and A the set of corresponding arcs.

[6] Other axioms can be generated lazily from stack and arcs set pairs of items created by DP steps.

words, we exploit the perfect matching induced by a derivation between shift and reduce transitions: if t_ℓ is a shift transition and $t_{\ell'}$ its matching reduction, we consider a score depending on the equivalence classes of c_ℓ and $c_{\ell'+1}$. Note that the nestedness property implies that the stacks σ_ℓ and $\sigma_{\ell'+1}$ are equal. Moreover, $i_{\ell'+1} = i_{\ell'}$ as $t_{\ell'}$ is a reduction, an operation which does not modify the buffer. Denoting by M the set of matching shift/reduce pairs $(t_\ell, t_{\ell'})$ in γ, this gives:

$$s^{(m)}(\gamma) = \sum_{(t_\ell, t_{\ell'}) \in M} s_m(i_\ell, |\sigma_\ell|, i_{\ell'}). \quad (4)$$

Note that $s_m(i_\ell, |\sigma_\ell|, i_{\ell'})$ can be seen as the score of performing a reduction when i_ℓ is on the top of the stack of size $|\sigma_\ell|$ and $i_{\ell'}$ is the first word position of the buffer. Hence, this gives a score similar to the reduction score used in the minimal feature set algorithm. The difference is that the score takes into account the size of the stack but not the type (left or right) of reduction that is performed. The direction information will be given by the dependency arc score.

Finally, the derivation score is:

$$s(\gamma) = s^{(u)}(\gamma) + s^{(m)}(\gamma) \quad (5)$$

3.2 Graph Representation of a Derivation

In this section we represent the derivation of the arc-hybrid parsing using graphs.

A derivation is a sequence of n shifts and n reductions such that no more reductions than shifts are performed at each step. Such a sequence is a Dyck word and can then be represented as a path in an $(n+1) \times (n+1)$ grid starting at the lower left corner, ending at the lower right corner, using only up-right diagonal arcs and downward arcs (Roman, 2015). Such representation is the starting point of our derivation graph.

Define the *derivation graph* $G = (V, A)$ as follows. $V = \{v_i^q | 1 \leq q \leq i \leq n+1\}$ represents the set of equivalence classes. A vertex v_i^q corresponds to the class (i, q) of configurations, where w_i is the first word in the buffer[7] and the stack is of size q. The arc set A is given by $A = T \cup E$ where T represents transitions between states and E matches between shifts and reductions. We note $T = T^S \cup T^L \cup T^R$:

- $T^S = \{(v_i^q, v_{i+1}^{q+1}) | 1 \leq q \leq i \leq n\}$,
- $T^L = \{(v_i^q, v_i^{q-1})^L | 2 \leq q \leq i \leq n\}$,
- $T^R = \{(v_i^q, v_i^{q-1})^R | 2 \leq q \leq i \leq n+1\}$.

T^S, T^L and T^R represent shifts and tagged reductions respectively. We set $E = \{(v_i^q, v_j^q) | 1 \leq q \leq i < j \leq n+1\}$ because a shift can be matched to a reduction only if the size of the stack is the same before the shift and after the reduce. Note that for a sentence with n words, G will have $O(n^2)$ vertices and $O(n^3)$ arcs.

A derivation γ obtained by the arc-hybrid parsing will be represented by a pair (P, M) where $P \subseteq T$ is a path representing the derivation γ and $M \subseteq E$ is the set of arcs matching the shifts with their associated reductions in the derivation.

More formally, (P, M) is a solution if and only if we have the following. P is a path from v_1^1 to v_{n+1}^1 in G using only arcs of T. By construction, it contains n arcs of T^S and n arcs of $T^L \cup T^R$. It then corresponds to the sequence of transitions $t_1, \ldots, t_{2n}$. Note that any arc (v_i^q, v_{i+1}^{q+1}) in P consists in pushing word w_i on the top of the stack and any arc (v_i^q, v_i^{q-1}) in P consists in popping the top of the stack. One can retrieve the sequence of configurations $c_1, \ldots, c_{2n+1}$ thanks to the transitions.

Remark that each vertex v_i^q covered by P corresponds to configuration c_{2i-q} since $i-1$ shifts and $i-q$ reduces have been performed. An arc (v_i^q, v_j^q) of E belongs to M if the transition t_{2i-q} is a shift, transition t_{2j-q-1} is a reduction and these shift and reduce operations are matched together. Hence, $(v_i^q, v_j^q) \in M$ implies that (v_i^q, v_{i+1}^{q+1}) and (v_j^{q+1}, v_j^q) belong to P.

One can associate a score s_u from Eq. (3) with each arc of T and a score s_m from Eq. (4) with each arc of E. In this case, $s(\gamma)$ corresponds to

$$s(\gamma) = \sum_{a \in P} s_u(a) + \sum_{a \in M} s_m(a) \quad (6)$$

As an illustration, the derivation presented in the introduction can be represented by the set of black arcs P and red arcs M in Figure 1.

3.3 Dynamic Programming Algorithm

From the graph representation above we can derive a DP algorithm which computes the score of the optimal solution subgraph. This algorithm can be seen as a specialization of the general framework presented in Section 2.2 for our scoring functions.

[7] The value $n+1$ for i indicates that the buffer is empty.

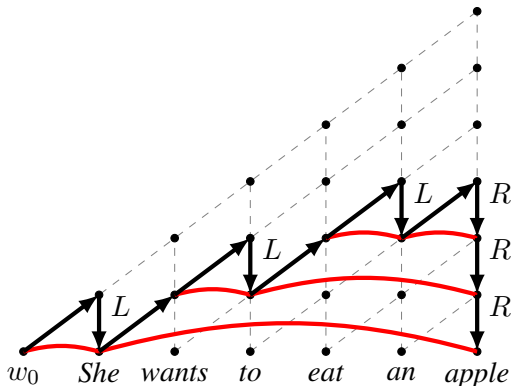


Figure 1: A Derivation for *She wants to eat an apple*. Shifts (resp. reductions) are represented in black diagonal (resp. vertical) arcs. Red curved edges represent matchings. Vertices are configuration classes v_i^q.

Since score functions s_u and s_m take the size of the stack into account, items with different (initial) stack sizes cannot be equivalent. Hence, items of the form ${}^q\langle \boldsymbol{i}, \boldsymbol{j} \rangle$ are needed, where $q = |\sigma|$ represents the size of the stack before shifting w_i. Leaving out the resulting dependency arcs, both *reduction* rules can be written with these equivalent items:

$$\frac{{}^q\langle \boldsymbol{i}, \boldsymbol{k} \rangle \quad {}^{q+1}\langle \boldsymbol{k}, \boldsymbol{j} \rangle}{{}^q\langle \boldsymbol{i}, \boldsymbol{j} \rangle}$$

There are $O(n^3)$ equivalent items and the DP complexity will be $O(n^4)$. Such items will be scored in the following way :

- the score of an axiom ${}^q\langle \boldsymbol{i}, \boldsymbol{i+1} \rangle$ is $s_u(i, q, \textit{shift})$. Axioms appear as black diagonal arcs (v_i^q, v_{i+1}^{q+1}) in Figure 1.
- In a DP step, the score of the *reduce* transition is $s_u(j, q+2, \textit{reduce})$. The transition is represented as a black vertical arc (v_j^{q+2}, v_j^{q+1}) in Figure 1.
- Finally the *matching* score in a DP step is $s_m(k, q+1, j)$. This score corresponds to red curved arcs (v_k^{q+1}, v_j^{q+1}) in Figure 1.

Figure 2 depicts both reduction rules. An item is represented as an arrow for the initial shift, and a triangle for the well-nested part of the subderivation. A reduction builds a new item by extending the well-nested part of the left antecedent with a new matching arc obtained from the right antecedent and the new reduction arc.

A dependency arc is added to assign a head to k, the midpoint of the reduction rule, depending on the direction of the reduction.

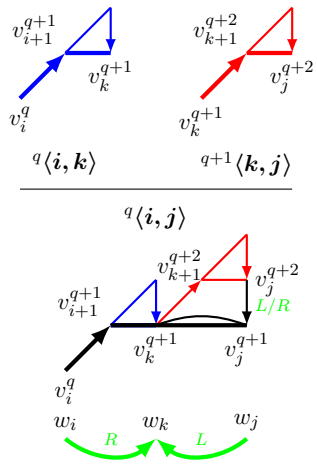


Figure 2: *Illustration of reduction rules.* Items are represented by a diagonal arc (first shift) and an horizontal edge (well-nested part). When combining two items in a reduction rule, a (curved) matching edge and a (vertical) reduction arc are added. The type of the reduction leads to a new dependency arc for the modifier w_k.

4 Learning Derivation Scores

We first present our network architecture inspired by recent work on self attention (Vaswani et al., 2017; Wang et al., 2017) which is able to learn representations of arc-hybrid configuration classes, that we call step representations. These representations are then used to compute derivation scores.

Our network is an encoder/decoder. The encoder computes word and step representations in the specific context of the sentence to be parsed, while the decoder computes arc scores.

We borrow from the transformer layer (Vaswani et al., 2017) the idea of global attention but extend it to the case where the size of the output is different from the size of the input. This has already been explored in (Gu et al., 2018) in the context of Machine Translation. However our problem is simpler because the size of the output is always twice the size of the input, in other words we do not have to estimate the size of the output.

4.1 Notation

A feed-forward layer is a sequence of an affine transformation, a ReLU filter and a linear transformation, *i.e.* $\text{FF}(\mathbf{x}) = V(\max(0, (W\mathbf{x} + b)))$, with V, W, b trainable parameters.

We call interpolation layers functions like $I(\mathbf{x}, \mathbf{y}) = C(\mathbf{x}) \cdot \mathbf{x} + (1 - C(\mathbf{x})) \cdot \mathbf{y}$ where C is a linear transformation followed by a sigmoid squashing, and $\cdot$ denotes the component-wise product.

Combining the previous two, we define highway layers (Greff et al., 2017) as functions $H(\mathbf{x}) = I(\mathbf{x}, FF(\mathbf{x}))$, i.e. an interpolation of input $\mathbf{x}$ and a feed-forward transformation of $\mathbf{x}$.

We also make use of biaffine functions following (Dozat and Manning, 2017) that we define as functions of the form $B(\mathbf{e}, \mathbf{f}) = \mathbf{e}^\top M \mathbf{f} + V\mathbf{e}$, with matrix M and vector V learnable parameters.

4.2 Layer Structure

Each layer L_i is the composition of two sublayers A_i, computing a generalized attention, and B_i, performing a feed-forward transformation. As is the case in previous approaches, each sublayer is followed by a layer normalization (Ba et al., 2016) and a residual connection to prevent underflows.

In more details, each layer takes as input a sequence of size n of dense vectors of size d packed as a matrix $\mathbf{X}$ in $\mathbb{R}^{n \times d}$ and a query vector sequence of size o, either equal to n or $2n$ depending on the layer (see infra) packed as a matrix $\mathbf{Y}$ in $\mathbb{R}^{o \times d}$. When $\mathbf{X}$ and $\mathbf{Y}$ are the same, we recover the self-attention mechanism of the transformer layer. The layer forward value is given by the following equations, where LN is a layer normalization:

$$\begin{aligned} a_i(\mathbf{X}, \mathbf{Y}) &= \mathbf{X} + A_i(\mathbf{X}, \mathbf{Y}) \\ b_i(\mathbf{X}) &= \mathbf{X} + B_i(\mathbf{X}) \\ L_i(\mathbf{X}, \mathbf{Y}) &= LN(b_i(LN(a_i(\mathbf{X}, \mathbf{Y})))). \end{aligned}$$

The first sublayer A_i computes for each output position in Y a multi-head scaled dot-product attention over input query Y and key/value X, with m attention heads.

$$A_i(\mathbf{X}, \mathbf{Y}) = \sum_{h=1}^{m} (A(Q_i^h \mathbf{Y}, K_i^h \mathbf{X}, V_i^h \mathbf{X}))$$

Each attention head A takes a query as input queries, keys and values, and computes for each query vector a convex combination of value vectors, the coefficients of which are given by an operation between the query and the key vectors:

$$A(\mathbf{Q}, \mathbf{V}, \mathbf{K}) = \text{softmax}(\mu \mathbf{Q}\mathbf{K}^\top)\mathbf{V}$$

where softmax is applied row-wise and μ is a smoothing factor between 0 and 1, which is set to $d^{-0.5}$, where d is the size of a query vector, following previous implementations of dot-product attention (Luong et al., 2015).

The second sublayer B_i applies the same feed-forward transformation to each element of the sequence of vectors returned by A_i.

4.3 Word and Position Embeddings

Each word in the train set is associated with a real vector stored in a lookup table E. In order to cope with unseen words, we follow (Kiperwasser and Goldberg, 2016) and at training time words are randomly `UNK`-ed with a probability inversely proportional to their frequency in the train set.

Contrarily to recurrent networks such as LSTMs, attention networks do not have a built-in notion of position so it must be provided externally. In our systems, we have two types of positions, namely word positions and step positions. We use position embeddings stored in lookup tables called respectively T and S.

4.4 Word Encoder and Dependency Scores

Our encoder is the composition of e self-attention layers starting from word and position vectors.

$$\begin{aligned} X_0 &= [E(w_1) + T(1); \ldots; E(w_n) + T(n)] \\ X_i &= L_i(X_{i-1}, X_{i-1}), 1 \leq i \leq e \end{aligned}$$

After encoding, we get X_e that we interpret as a sequence of contextualized word vectors. We can use these vectors to predict arc scores. In the following, we use a biaffine function B_{dep} to define the raw score between head at position i and modifier at position j: $s_{w_i \to w_j} = B_{\text{dep}}(X_e[i], X_e[j])$.

These raw scores are used in two ways.First and most obviously they score dependency arcs of the derived tree in this model. Second, for each modifier we use incoming arc scores to weigh potential heads and compute an expected head vector. We use normalized scores via softmax to interpolate head vectors. As a result for each vector word $X_e[i]$ we obtain an expected head vector $Y_e[i]$, which will be used hereafter.

4.5 Step Encoder

For steps, we distinguish the first layer from the others. For the first layer, input is the sequence returned by the last word encoder noted X_e and expected heads Y_e, and the query is initialized with

the increasing sequence of valid step position embeddings called $P = [S(1); \ldots; S(2n)]$.

$$D_0 = X_e + Y_e,\ P_1 = P,\ D_1 = L_{e+1}(D_0, P_1)$$

For $k > 1$, we set $D_k = L_{e+k}(D_{k-1}, D_{k-1})$.

Given sentence w, we note $h(w)$ the sequence of vectors returned by the last decoder layer.

4.6 Decoders as Local Classifiers

In this section we present how the score function can be decomposed as local probabilities. Given a sentence w, we assume the probability of a derivation γ is conditioned upon its corresponding derived tree τ. This condition prevents inconsistencies between τ and γ but plays no other role in the scoring function. The probability of a derived tree decomposes as independent head predictions computed by a logistic regression over head scores.

$$\begin{aligned} p(\gamma, \tau|w) =& p(\gamma|w, \tau) \times p(\tau|w) \\ =& p(\gamma|w, \tau) \times \prod_{h \to m \in \tau} p(h|m, w) \end{aligned}$$

The probability of a derivation in D_w is the probability at each independent step ℓ of 2 events: the transition t_ℓ and the step position r_ℓ of the corresponding reduction for a shift, or a dummy position for a reduction. We consider these two events to be independent but requiring the knowledge of the difference q_ℓ between the number of shifts and reductions already performed before the current step. This difference can also be interpreted in the context of the arc-hybrid algorithm as the depth of the stack. The difference is then used as a parameter for the potentials. This gives:

$$\begin{aligned} p(\gamma|w) &= \prod_{\ell=1}^{2|w|} p(t_\ell, r_\ell, q_\ell|w, \ell) \\ &= \prod_{\ell=1}^{2|w|} p_d(q_\ell|w, \ell) \times p(t_\ell, r_\ell|w, \ell, q_\ell) \\ &= \prod_{\ell=1}^{2|w|} p_d(q_\ell|w, \ell) \\ &\qquad \times p_u(t_\ell|w, \ell, q_\ell) \times p_m(r_\ell|w, \ell, q_\ell) \end{aligned}$$

The condition on sentence and step index w, ℓ is implemented via functions taking the ℓ^{th} step representation of w computed as described in the previous section, denoted $h(w)_\ell$ or simply h_ℓ if the sentence is clear from the context.

In practice we restrict the set of values for q_ℓ as the set of natural numbers between one and nine, and a special value for differences greater or equal to ten[8]. We use a lookup table F to convert discrete difference values to dense representations.

Note that although the 3 distributions are conditioned on the same step representations introduced in the previous section, these representations are first passed through highway layers, $\{H_i\}_{i=d,u,m}$, parameterized for each distribution. This helps with the specialization of step representations while keeping the possibility to share information between tasks.

The first two distributions are categorical distributions of the exponential family. They are computed as normalized potentials given by feedforward transformations of steps and differences.

$$\begin{aligned} p_d(q|h_\ell) &\propto \exp \mathrm{FF}_d(F(q) + H_d(h_\ell)), \\ p_u(t|q, h_\ell) &\propto \exp \mathrm{FF}_t(F(q) + H_u(h_\ell)). \end{aligned}$$

The third distribution is computed with a biaffine function B_m followed by a softmax, as in (Dozat and Manning, 2017). We reserve an extra value for the result random variable which encodes the absence of corresponding reduction.This is used when s is not a shift step. Note the difference embedding is only used on the left side of M.

$$p_m(r|h_\ell, q) \propto \exp B_m(F(q) + H_m(h_\ell), h_{\ell+r})$$

Learning is performed by simply maximizing the conditional log-likelihood of the 4 distributions over the correct derivations given a set of sentences. Once parameterized, it is straightforward to see how these distributions can fit the score model of Equation 5.

Conditional log-likelihood minimization requires to compute values for each distribution along gold solutions, which means it is a $O(n^2)$ procedure, because of distributions on dependency arcs and matching transitions, which both require 2 position parameters in order to be computed.

5 Experiments

Data We ran experiments on the Wall Street sections of the English Penn Treebank (Marcus

[8] We found almost all train sentences could be parsed with stack size below 10.

POS embedding size	28
Other embedding size	100
Encoder input/output size	256
Decoder input/output size	256
Hidden layer size in B subnetworks	512
attention heads	8
Maximum step numbers	200
Maximum difference D (stack height)	10
Number of word encoder layers	4
Number of step encode layers	4

Table 1: Network hyperparameters

Setting	UAS
Ours	$\mathbf{94.82} \pm 0.10$
Ours, joint training, but decoding with	
dependency scores only	94.80 ± 0.09
derivation scores only	93.95
Ours, training and decoding with	
dependency scores only	94.73 ± 0.06
derivation scores only (no stack size)	84.81
(Shi et al., 2017) best local (4 features)	93.89
(Shi et al., 2017) global (2 features)	94.43
(Shi et al., 2017) global Eisner	94.50
(Kiperwasser and Goldberg, 2016) greedy	93.8

Table 2: Comparisons on PTB test set

et al., 1994) converted to Stanford Dependencies (de Marneffe and Manning, 2008). Transition sequences were obtained from dependencies and in case of ambiguity right reductions were always performed before shift if possible. We followed the standard split (02-21 for training, 23 for testing and 22 for development purposes) and used POS tags predicted by the Stanford MaxEnt tagger trained using 10-way jackknifing (Toutanova et al., 2003). We evaluate on unlabeled attachment without punctuation, with CoNLL evaluation script.

Implementation Hyperparameters are given in Table 1. In addition to word embeddings parameterized on the PTB, we use pretrained Glove vectors (Pennington et al., 2014). Our prototype is written in C++ with DYNET[9] for neural computations (Adam optimizer and default values) and UDPIPE[10] for reading and writing data files. Minibatches contain around 1,000 tokens. We train each model for 100 epochs and select our best model according to its development UAS. We follow previous works with transformers to set the learning rate (Vaswani et al., 2017). For the first $8,000$ updates the learning increases linearly with the number of steps, then it decreases proportionally to the squared root of the number of steps. We use the formula of Strubell et al. (2018).

Results Results on the PTB test set are presented in Table 2 and comparisons with previous work on arc-hybrid parsing with comparable network sizes. Parsing results are obtained by averaging 5 models initialized with different random seeds and standard deviation is also provided. Our system reaches 94.8% UAS and parses the whole section 23 in 1.13 seconds (DP only) on an Intel Xeon 2.10 GHz. Although our system is trained via local classifiers, we can see that it improves over global systems trained without step encodings.

Ablations indicate that the major part of scoring comes from dependencies. We may also conclude that (i) derivation information is useful per se but most importantly as an auxiliary task to improve dependencies and (ii) the stack size is paramount in this model since otherwise the network has no indication of the stack content. If not considered, accuracy drop considerably: step indexes alone are too vague as they can correspond to many different stack and buffer contents.

6 Discussion and Related Work

Derivation Parsing Maximum subgraph selection has played a central role in dependency parsing since the MST reduction by McDonald et al. (2005) and can also be traced back to the parsing-as-intersection tradition in phrase-based parsing – see for instance (Billot and Lang, 1989) – where the goal is to find, starting from a generic grammar, a graph-structure (a shared forest) that recognizes the input presented as a string or an automaton. In dependency parsing, this approach has since been extended to more complex dependencies such as non-crossing and 1-endpoint-crossing dependencies (Kuhlmann and Jonsson, 2015; Cao et al., 2017).

There is a long line of research which solve the different variants of transition-based dependency parsing algorithms with dynamic programming. Recent work showed that this can be performed efficiently, in $O(n^3)$, for arc-hybrid parsers (Gómez-Rodríguez et al., 2008) and have since been extended with non-linear classifiers (Kiperwasser and Goldberg, 2016; Shi et al., 2017) to reach state-of-the-art parsing accuracy.

We depart from both in the following way. In most works on maximum subgraph selection, the class of valid subgraph is a class defined only by

[9] https://github.com/clab/dynet
[10] http://ufal.mff.cuni.cz/udpipe

properties on dependencies. Here we represent *derivations* instead of derived structures. In that respect, we are closer to approaches developed for mildly-context sensitive formalisms such as Tree Adjoining Grammars which work primarily on the derivation tree (Corro et al., 2017) and consider the derived tree, i.e. the parse structure, as a by-product. Compared to other dynamic programming approaches to arc-hybrid parsing, we therefore work on a richer model, and have more expressive power to take a representation of states into account in the scoring scheme. This comes at a cost since the time complexity of our parsing algorithm is $O(n^4)$, an order of magnitude higher. The stack information (size) is minimal and is used to parameterize access to information available from step embeddings.

Compared to joint parsing systems working on both constituents and dependencies, our approach doesn't require external linguistic knowledge such as head percolation rules. On the other hand, since derivations don't add new information, but merely offer a new vision of the problem, the potential accuracy gain is lower.

Machine Learning Aspects Self-attention networks have been used in parsing, see for instance (Kitaev and Klein, 2018), whether based on dependencies or syntagms. Curiously we found few models of transition-based parsing based on these networks, and bidirectional recurrent network are still preferred in most architectures, where they are believed to capture some information about the sequential nature of transition-based algorithms. Instead we present a non-sequential model of transition-based parsing where representation vectors are obtained via unrolled iterative estimation (Greff et al., 2017).

Our encoder-decoder architecture together with independence assumptions made in the probabilistic model which decomposes a derivation score in several subtasks can be seen as auxiliary tasks as in (Coavoux et al., 2018).

The use of expected head vectors as input of the step encoder is related to the syntactic head attention of the SRL neural architecture in (Strubell et al., 2018).

7 Conclusion

We presented the arc-hybrid parsing transition rule system as a subgraph selection problem and showed how this can be solved exactly by a dynamic programming algorithm. This theoretical result is backed up by state-of-the-art results on the PTB.

This new representation of the problem is the basis of a novel neural architecture which learns vertex representations (for derivation steps) and edge scores (for derivation features).

From a parsing perspective, understanding why derivation prediction is a good auxiliary task to learn syntactic dependencies could prove insightful to explain how transitions and dependencies are related.

The derivation/derived pair is a very powerful concept that remains to be fully exploited. In particular, there are two promising avenues for future improvements. First, the learning framework could be enriched in a setting where the derivation graph is modeled as a latent variable and marginalized over. It remains to be seen if this can be done exactly or if sampling is required for efficiency. Second, since the score of a solution is the sum of the scores of elements in a pair, it should be possible to design an approximate solver based on lagrangian decomposition more efficient in practice.

Acknowledgments

This work is partially supported by a public grant overseen by the French National Research Agency (ANR) as part of the program *Investissements d'Avenir* (ANR-10-LABX-0083). It contributes to the IdEx Université de Paris (ANR-18-IDEX-0001). This work is partially supported by a public grant overseen by the French ANR (ANR-16-CE33-0021).

References

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Sylvie Billot and Bernard Lang. 1989. The structure of shared forests in ambiguous parsing. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pages 143–151. Association for Computational Linguistics.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017. Quasi-second-order parsing for 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 24–34. Association for Computational Linguistics.

Maximin Coavoux, Shashi Narayan, and Shay B. Cohen. 2018. Privacy-preserving neural representa-

tions of text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1–10. Association for Computational Linguistics.

Caio Corro, Joseph Le Roux, and Mathieu Lacroix. 2017. Efficient discontinuous phrase-structure parsing via the generalized maximum spanning arborescence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1644–1654. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *International Conference on Learning Representations*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Carlos Gómez-Rodríguez, John Carroll, and David Weir. 2008. A deductive approach to dependency parsing. In *Proceedings of ACL-08: HLT*, pages 968–976, Columbus, Ohio. Association for Computational Linguistics.

Klaus Greff, Rupesh Kumar Srivastava, and Jürgen Schmidhuber. 2017. Highway and residual networks learn unrolled iterative estimation. In *International Conference on Learning Representations*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686. Association for Computational Linguistics.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 673–682, Portland, Oregon, USA. Association for Computational Linguistics.

Marco Kuhlmann and Peter Jonsson. 2015. Parsing to noncrossing dependency graphs. *Transactions of the Association for Computational Linguistics*, 3:559–570.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421. Association for Computational Linguistics.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *HLT'94: Proceedings of the workshop on Human Language Technology*, pages 114–119, Morristown, NJ, USA. Association for Computational Linguistics.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. Stanford typed dependencies manual. Technical report, Stanford University.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Steven Roman. 2015. *An Introduction to Catalan Numbers*, 1st edition. Birkhäuser Basel.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 12–23. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038. Association for Computational Linguistics.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. 2017. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 10.

Policy Preference Detection in Parliamentary Debate Motions

Gavin Abercrombie
Department of Computer Science
University of Manchester
gavin.abercrombie@
manchester.ac.uk

Federico Nanni
Data and Web Science Group
University of Mannheim
federico@
informatik.uni-mannheim.de

Riza Batista-Navarro
Department of Computer Science
University of Manchester
riza.batista@
manchester.ac.uk

Simone Paolo Ponzetto
Data and Web Science Group
University of Mannheim
simone@
informatik.uni-mannheim.de

Abstract

Debate *motions* (proposals) tabled in the UK Parliament contain information about the stated policy preferences of the Members of Parliament who propose them, and are key to the analysis of all subsequent speeches given in response to them. We attempt to automatically label debate motions with codes from a pre-existing coding scheme developed by political scientists for the annotation and analysis of political parties' manifestos. We develop annotation guidelines for the task of applying these codes to debate motions at two levels of granularity and produce a dataset of manually labelled examples. We evaluate the annotation process and the reliability and utility of the labelling scheme, finding that inter-annotator agreement is comparable with that of other studies conducted on manifesto data. Moreover, we test a variety of ways of automatically labelling motions with the codes, ranging from similarity matching to neural classification methods, and evaluate them against the gold standard labels. From these experiments, we note that established supervised baselines are not always able to improve over simple lexical heuristics. At the same time, we detect a clear and evident benefit when employing BERT, a state-of-the-art deep language representation model, even in classification scenarios with over 30 different labels and limited amounts of training data.

1 Introduction

Commonly known as the *Hansard* record, transcripts of debates that take place in the House of Commons of the United Kingdom (UK) Parliament are of interest to scholars of political science as well as the media and members of the public who wish to monitor the actions of their elected representatives. Debate *motions* (the proposals tabled for debate) are expressions of the policy positions taken by the governments, political parties, and individual Members of Parliament (MPs) who propose them. As all speeches given and all votes cast in the House are responses to one of these proposals, the motions are key to any understanding and analysis of the opinions and positions expressed in the subsequent speeches given in parliamentary debates.

By definition, debate motions convey the stated policy preferences of the MPs or parties who propose them. They therefore express polarity—*positive* or *negative*—towards some target, such as a piece of legislation, policy, or state of affairs. As noted by Thomas et al. (2006), the polarity of a debate proposal can strongly affect the language used by debate participants to either support or oppose it, effectively acting as a polarity shifter on the ensuing speeches. Analysis of debate motions is therefore a key first step in automatically determining the positions presented and opinions expressed by all speakers in the wider debates.

Additionally, there are further challenges associated with this task that differentiate it from the forms of sentiment analysis typically performed in other domains. Under Parliament's *Rules of Behaviour*,[1] debate participants use an esoteric speaking style that is not only laden with opaque procedural language and parliamentary jargon, but is also indirect, containing few explicitly negative words or phrases, even where negative positions are being expressed (Abercrombie and Batista-Navarro, 2018a).

The topics discussed in these debates revolve

[1] https://www.parliament.uk/documents/rules-of-behaviour.pdf

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 249–259
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

around policies and policy domains. Topic modelling or detection methods, which tend to produce coarse overviews and output neutral topics such as 'education' or 'transport' (as in Menini et al. (2017), for instance), are therefore not suitable for our purposes. Rather, we seek to find the proposer of a motion's position or *policy preference* towards each topic—in other words, an *opinion-topic*. Topic labels do exist for the Hansard transcripts, such as those produced by the House of Commons Library or parliamentary monitoring organsitions such as Public Whip.[2] However, these are unsuitable due to, in the former case, the fact that they incorporate no opinion or policy preference information, and for the latter, being unsystematic, insufficient in both quantity and coverage of the topics that appear in Hansard, and not future-proof (that is, they do not cover unseen topics that may arise (Abercrombie and Batista-Navarro, 2018b)).

In this paper, we use the coding scheme devised by the Manifesto Project,[3] because: *(a)* it is systematic, having been developed by political scientists over a 40 year period, *(b)* it is comprehensive and designed to cover any policy preference that may be expressed by any political party in the world, *(c)* it has been devised to cover any policies that may arise in the future, and (d) there exist many expert-coded examples of manifestos, which we can use as reference documents and/or for validation purposes.

We approach automatic policy preference labelling at both the motion and (quasi-)sentence levels (see Section 2). We envisage that the output could therefore be used for downstream tasks, such as sentiment and stance analysis and agreement assessment of debate speeches, which may be performed at different levels of granularity.

Our contributions This paper makes the following contributions to the literature surrounding natural language processing of political documents and civic technology applications:

1. We develop a corpus of English language debate motions from the UK Parliament, annotated with policy position labels at two levels of granularity. We also produce annotation guidelines for this task, analysis of inter-annotator agreement rates, and further evaluation of the difficulty of the task on data from both parliamentary debates and the manifestos. We make these resources publicly available for the research community.

2. We test and evaluate two different ways of automatically labelling debate motions with Manifesto Project codes: lexical similarity matching and supervised classification. For the former, we compare a baseline of unigram overlap with cosine similarity measurement of vector representations of the texts. For the latter, we test a range of established baselines and state-of-the-art deep learning methods.

2 Background

Rather than being forums in which speakers attempt to persuade one another of their points of view, as the word 'debate' may imply, parliamentary speeches are displays of position-taking that MPs use to communicate their policy preferences to 'other members within their own party, to members of other parties, and, most important, to their voters' (Proksch and Slapin, 2015). Debate *motions* are proposals put forward in Parliament, and as such are the objects of all votes and decisions made by MPs, and, in theory at least, of all speeches and utterances made in the House.[4] Each parliamentary debate begins with such a motion, and may include further *amendment* motions (usually designed to alter or reverse the meaning of the original) as it progresses. Motions routinely begin with the words '*I beg to move That this House ...*', and may include multiple parts, as in *Example 1*,[5] which consists of two clauses, and appears to take a positive position towards international peace:

> *I beg to move*
> *That this House notes the worsening humanitarian crisis in Yemen;*
> *and calls upon the Government to take a lead in passing a resolution at the UN Security Council that would give effect to an immediate ceasefire in Yemen.* (1)

The concept of *policy preferences* is widely used in the political science literature (e.g. Budge

[2] https://www.publicwhip.org.uk
[3] https://manifestoproject.wzb.eu
[4] https://www.parliament.uk/site-information/glossary/motion
[5] https://hansard.parliament.uk/commons/2017-03-28/debates/F81005F8-5593-49F8-82F7-7A62CB62394A/Yemen

et al., 2001; Lowe et al., 2011; Volkens et al., 2013) to represent the positions of political actors expressed in text or speech. The Manifesto Project is an ongoing venture that spans four decades of work in this area and consists of a collection of party political documents annotated by trained experts with codes (labels) representing such preferences. Organised under seven 'domains', the coding scheme comprises 57 policy preference codes, all but one of which (*408: Economic goals*) are 'positional', encoding a positive or negative position towards a policy issue (Mikhaylov et al., 2008). Indeed, many of these codes exist in polar opposite pairs, such as *504: Welfare State Expansion* and *505: Welfare State Limitation*. The included manifestos are coded at the *quasi-sentence* level—that is, units of text that span a sentence or part of a sentence, and which have been judged by the annotators to contain 'exactly one statement or "message"' (Werner et al., 2011), as in *Example 2*, in which a single sentence has been annotated as four quasi-sentences:[6]

> *To secure your first job we will create 3 million new apprenticeships;*
> 411: Technology and Infrastructure
>
> *take everyone earning less than 12,500 out of Income Tax altogether*
> 404: Economic Planning
>
> *and pass a law to ensure we have a Tax-Free Minimum Wage in this country;*
> 412: Controlled Economy
>
> *and continue to create a fairer welfare system where benefits are capped to the level that makes work pay so you are rewarded for working hard and doing the right thing.*
> 505: Welfare State Limitation

(2)

3 Related work

There exists a large body of work concerning the analysis of opinions and policy positions in the related domains of legislative debate transcripts (for a survey, see Abercrombie and Batista-Navarro, 2019) and party political manifestos (see Volkens et al., 2015). Inspired by work on analysis of text from other domains, such as product reviews and social media, much of the computer science research in this area has concentrated on classifying the sentiment polarity of individual speeches (e.g. Burford et al., 2015; Thomas et al., 2006; Yogatama et al., 2015). Political scientists meanwhile, have tended to focus on position scaling—the task of placing the combined contributions of a political actor on a (usually) one-dimensional scale, such as *Left–Right* (e.g. Glavaš et al., 2017b; Laver et al., 2003; Nanni et al., 2019a; Proksch and Slapin, 2010). In either case, the majority of this work does not take into consideration the topics or policy areas addressed in the speeches.

Supervised classification approaches to opinion-topic identification have been explored in a number of papers. Abercrombie and Batista-Navarro (2018b) obtain good performance in classifying debate motions as belonging to one of 13 'policies' or opinion-topics. However, this approach is somewhat limited in that they use a set of pre-existing labelled examples which does not extend to cover the whole Hansard corpus or any new policies that may arise in the future. A similar setting to ours is that of Herzog et al. (2018), who use labels from the Comparative Agendas Project (CAP).[7] However, while they seek to discover latent topics present in the corpus, we wish to determine the policy-topic of each individual debate/motion. Rather than employ labelled manifesto data, as we do, they use the descriptions of the CAP codes.

Concerning policy identification in party political manifestos, previous studies have focused on topical segmentation (Glavaš et al., 2016) and classification of sentences into the seven coarse-grained policy domains (Glavaš et al., 2017a; Zirn et al., 2016). Meanwhile, Subramanian et al. (2018) recently presented a deep learning model that classifies manifesto sentences with the finer-grained code-level scheme of the Manifesto Project, as well as placing them on a Left-Right scale. In order to contribute to these research efforts and following recent advancements in deep language representation models (Devlin et al., 2018; Peters et al., 2018), we test the potential of BERT (Bidirectional Encoder Representations from Transformers) for policy-topic classification on both debate motions and manifestos.

[6] Conservative Party manifesto 2015.

[7] https://www.comparativeagendas.net

There is also a growing body of research on the evaluation of annotations for this domain. While the Manifesto Project relies on trained individual annotators to label manifestos, Mikhaylov et al. (2008) report the results of experiments which show that agreement between annotators is difficult to achieve, casting doubts on the reliability of the Project's codes. However, in similar experiments, Lacewell and Werner (2013) report greater inter-annotator agreement, and claim that with ongoing training, annotators can produce reliable labels. An extended analysis of the validity and reproducibility of the coding scheme is offered by Gemenis (2013), who remarks on the fact that 'the problem of unreliability does not lie with the coders but with the complex nature of the CMP (Comparative Manifesto Project) coding scheme'. Aware of such challenges, and in order to offer an additional comparison to these previous studies, in this work we provide a detailed analysis of the agreement rates of our annotators on both manifestos and debate motions.

4 Data

In the experimental section we report on the use of codes from the Manifesto Project as policy preference labels, with the goal of applying them to debate motions. These labels are convenient because: *(a)* like debate transcripts, they have been collected over time; and *(b)* the Project is ongoing, meaning that new example manifestos will continue to be added to it, mitigating potential concept drift problems (in which the language used to refer to aspects of different policy areas may change diachronically).

To construct our corpus, we made use of the data sources described below:

The Manifesto Project

We used annotated manifestos *(1)* as reference texts for labelling of debate motions by similarity matching, and *(2)* training a neural network for cross-domain classification of the motions. We downloaded all fifteen of the annotated United Kingdom (including Northern Ireland) manifestos from the Manifesto Corpus Version 2018-1 (Krause et al., 2018)—that is those that have been coded under version 4 of the coding scheme.[8]

[8] https://manifestoproject.wzb.eu/coding_schemes/mp_v4

Domain	Manifestos	Debates	
	QSs	Motions	QSs
1: External Relations	1,436	50	186
2: Freedom & Democracy	767	30	106
3: Political System	1,627	47	220
4: Economy	4,296	87	380
5: Welfare & Quality of Life	2,235	118	528
6: Fabric of Society	1,574	33	153
7: Social Groups	1,180	21	110
0: No meaningful category	166	0	0

Table 1: The number of quasi-sentences (QSs) coded under each domain in the UK manifestos that we use as reference texts and training data and the number of debate motions and quasi-sentences that we label under each domain in the motion policy preference corpus.

Party	Year(s)	QSs
Conservative	2015	1589
DUP	2015	229
Green Party	2015	2235
Labour	2001, 2015	2503
Liberal Democrats	1997, 2015	2759
Plaid Cymru	2015	776
SDLP	2015	407
Sinn Féin	2015	272
SNP	1997, 2001, 2015	2309
UKIP	2015	1349
UUP	2015	417

Table 2: The parties and years of publication of the manifestos that we use as reference texts and training data, and the number of labelled quasi-sentences (QSs) by party in this subset of the manifesto data.

In this subset, the number of UK manifesto quasi-sentences labelled with codes in each domain varies considerably (see Table 1). These manifestos were written by a variety of political parties for elections over an 18 year period (Table 2). The most prevalent code in these manifestos is *504: Welfare State Expansion* (2,691 examples), and the least used is *103: Anti-Imperialism* (3 examples). Two codes, *102: Foreign Special Rela-*

tionships: Negative and *415: Marxist Analysis: Positive*, do not appear at all in manifestos from the United Kingdom.

Debate transcripts

The Hansard record of House of Commons debates is available for each day on which debates have taken place from 1919 to the present day in `xml` format at `https://www.theyworkforyou.com`, where it is updated daily with the most recent debates. As the record is more complete for recent years, we downloaded all files from May 7th 1997 (the start of that year's session of Parliament) to February 28th 2019. From these we extracted 1,156 motions together with the titles of the debates and the dates on which they were tabled. We manually removed procedural motions (those concerned solely with the workings of Parliament) from the dataset as these do not concern policy preferences and have no equivalents in political manifestos.

In order to approximate the format of the data in the Manifesto Project, and to investigate policy preference detection at different levels of granularity, we divided each motion into smaller units. For convenience, we approximated quasi-sentences in the Hansard data by automatically dividing motions into clauses, which are separated by semi-colons in the transcripts.

5 Annotation

We adapt the Project's *Coding Instructions* (Werner et al., 2011) to provide guidelines for the annotation of debate motions. We use version 4 of these instructions because, although a more recent, more finely grained version exists, there are as yet few example manifestos coded under the newer scheme. To complete the annotation task, we recruited three Political Science Master's students from the University of Mannheim, who worked for a total of 40 hours each over a two month period.

Debate motions

Annotations were carried out in two stages: an initial training phase, followed by labelling of the main dataset. We used the coding instructions of version 4 of the Manifesto Project handbook[9] supplemented by debate motion-specific guidelines including notes based on the annotators' discussions during training.[10] For the training phase, after being introduced to the data and the coding instructions, the annotators individually labelled three batches of motions and their quasi-sentences. In addition to labelling each of these with one of the codes, they were instructed to note examples which they found difficult to decide upon. Between each batch we met to discuss these instances, as well as other examples on which the annotators disagreed, adding notes to the annotation guidelines based on the observations made. Inter-annotator agreement during training ranged from 'fair' to 'substantial', following common interpretation of Fleiss' *kappa* scores (Landis and Koch, 1977) (see Table 3).

The final corpus includes 386 hand-annotated motions and 1,683 quasi-sentences.[11] The majority of these have been labelled by two of the three annotators. Inter-annotator agreement is within the ranges generally interpreted as being 'moderate' to 'substantial' (see Table 4). The slightly higher agreement at the quasi-sentence level than on overall motion labels suggests that it may be difficult in some cases to select a single policy preference code for a whole motion. A subsection of the corpus (41 motions, 180 quasi-sentences) was labelled by all three annotators. Fleiss' *kappa* scores for this subsection are 0.46 at both levels, which indicates 'moderate' agreement. Following Pustejovsky and Stubbs (2012), the gold standard label for each example is obtained by adjudication, which was carried out by the first author.

Manifestos

To validate our labelling procedure, and for comparison with other work, we also asked the annotators to label a small quantity (120) of quasi-sentences from the Manifesto Project. We calculate Fleiss' *kappa* for these annotations to be 0.48, which is comparable to that obtained on the main dataset of debate motions, and higher than those reported by Mikhaylov et al. (2008) on manifestos.

Again, we asked the annotators to mark any examples which they considered to be difficult to decide upon. Agreement (Fleiss' *kappa*) on these 'difficult' cases is only 0.17, with only one ex-

[9] Available at `https://manifestoproject.wzb.eu/down/papers/handbook_2011_version_4.pdf`

[10] These guidelines are available along with the corpus.

[11] These constitute examples with 'gold standard' labels. The corpus also includes examples labelled by a sole annotator ('silver standard') and further unlabelled motions (see Table 5).

Iteration	Motion level			Quasi-sentence level		
	No. of examples	*k*	Interpretation	No. of examples	*k*	Interpretation
Training 1	15	0.41	'moderate'	60	0.35	'fair'
Training 2	12	0.65	'substantial'	60	0.56	'moderate'
Training 3	16	0.48	'moderate'	60	0.40	'fair'

Table 3: Annotator agreement (Fleiss's *kappa*) at two levels of granularity during three iterations of training and development of annotation guidelines for labelling debate motions with codes from the Manifesto Project.

	Annotators	No.	*k*
Motion	All 3	41	0.46
QS	All 3	180	0.46
Motion	1 & 2	139	0.51
	2 & 3	155	0.50
	1 & 3	169	0.49
	All pairs	463	0.50
QS	1 & 2	622	0.58
	2 & 3	650	0.51
	1 & 3	731	0.62
	All pairs	2003	0.58

Table 4: Fleiss' *kappa* scores for three-way agreement and Cohen's *kappa* scores for two-way agreement on the debate motions dataset.

Labels	Motion	Quasi-sentence
Gold standard	386	1,683
Silver standard	87	361
Total labelled	473	2,044
Unlabelled	593	2,587
Overall total	1,066	4,631

Table 5: Statistics for the motion policy preference corpus. Gold standard examples have been labelled by two or three annotators initially and adjudicated on in a final round of annotation. Silver standard examples have been labelled by a single annotator only.

ample marked as such by all three annotators. In this case, two of them used the 'correct' Manifesto Project gold label, while the third annotator applied a different code from the same domain. Overall, of the 47 examples (39.2%) on which all three annotators agree, 36 of these agree with the gold label (30% of the total). Domain-level agreement is 0.56, which is also similar to that achieved on the debate motions.

The Motion Policy Preference Corpus

We make the corpus available for download at https://madata.bib.uni-mannheim.de/308. The number of labelled and unlabelled examples it contains can be seen in Table 5. For the gold-labelled data, motions range in length from one to 13 quasi-sentences (mean = 4.3), with each of these consisting of between four and 163 tokens (mean = 28.7).

6 Automatic Labelling Methods

We investigated two ways of automatically labelling debate motions with the codes from the Manifesto Project: *(1)* similarity matching and *(2)* supervised classification. We tested both at the quasi-sentence level and we additionally experiment with similarity matching methods at the whole motion level, where the lack of sufficient training data prevents application of supervised learning methods. In pre-processing we filtered out any motions that have gold standard labels that appear less than ten times in the corpus, leaving 370 motions and 1,634 quasi-sentences, each annotated with one of the 32 remaining class labels.

Similarity matching

We tested two methods of matching debate motions to codes from the Manifesto Project, comparing a baseline of unigram overlap scores with cosine similarity measurement. In each case, we measured the similarity of the list of tokens $A = A_1, A_2, ..., A_n$ in each motion or quasi-sentence text and the list of tokens in each collection of concatenated manifesto extracts $B = B_1, B_2, ...B_n$.

For unigram overlap, we simply counted the union of the sets of tokens from *A* and *B*. For the latter method, each text was represented by its term frequency-inverse document frequency vector (tf-idf), and cosine similarity calculated as:

$$\frac{\vec{A} \cdot \vec{B}}{||\vec{A}||||\vec{B}||}$$

With both of these approaches, we explored the use of the following combinations of sources of textual unigram features: the debate titles, which have been shown to be highly predictive of a

motion's opinion-topic in a supervised classification setting (Abercrombie and Batista-Navarro, 2018b), the debate motions themselves, and both the titles and motions together.

Supervised Classification

We tested a range of supervised machine learning algorithms for the policy preference classification task, ranging from traditional approaches to recently developed pre-trained deep language representation models. We were particularly interested in assessing the performance of such approaches: *(1)* despite the limited training data available (1.6k motion quasi-sentences); and *(2)* in a cross-domain application (training on over 16k manifesto quasi-sentences, and testing on the motion quasi-sentences).

First, we examined the performance of Support Vector Machines (SVM) trained using lexical (tf-idf) or word embedding (w-emb) features, which act as strong traditional baselines. We tested both pre-trained general purpose word embeddings from `https://fasttext.cc` (Mikolov et al., 2018) and in-domain vectors generated on the Hansard transcripts from Nanni et al. (2019b).

We also report the results of a widely adopted neural network baseline for topic classification (see for instance Glavaš et al. (2017a) and Subramanian et al. (2018) in the context of manifesto quasi-sentences classification): a Convolutional Neural Network (CNN) with single convolution layer and a single max-pooling layer. We again tested the CNN with general purpose and in-domain embeddings.

As final skyline comparisons, we present the performance of *(1)* a pre-trained BERT (large, cased) model (Devlin et al., 2018), with a final soft-max layer; and *(2)* the same pre-trained BERT model, with a CNN and max-pooling layers before the soft-max layer. We additionally experimented with the latter two models in a fine-tuning setting: after training on manifestos, they have been further fine-tuned on motions.

We tested all approaches with a 80/20 split of the dataset, and trained all the neural models for three iterations.

7 Results

We evaluated the predicted labels of each experimental model against the gold standard labels produced by the annotation process. For the machine learning methods, we report F1 scores with both macro and micro weightings in order to offer an understanding of the quality overall, as well as for the different classes.

Motions: Similarity Matching

We evaluate labelling of motions by similarity matching at two levels of granularity: quasi-sentence and whole motion. Cosine similarity matching comfortably outperforms the baseline at both levels of granularity and at both the policy and domain levels (see Figure 1).

Unlike the findings of Abercrombie and Batista-Navarro (2018b), in most settings, we do not find the debate titles to be as powerful indicators of class labels as features derived from the texts of the motions, perhaps due to our larger set of class labels containing more similar (same domain) policy preference codes.

Best performances at both policy and domain levels (F1 macro = 0.59) are obtained using tf-idf features derived from both motion titles and texts, although performance using the texts only is comparable. For most combinations of feature input and similarity measurement method, F1 scores are around twice as good at the domain level as at the policy level.

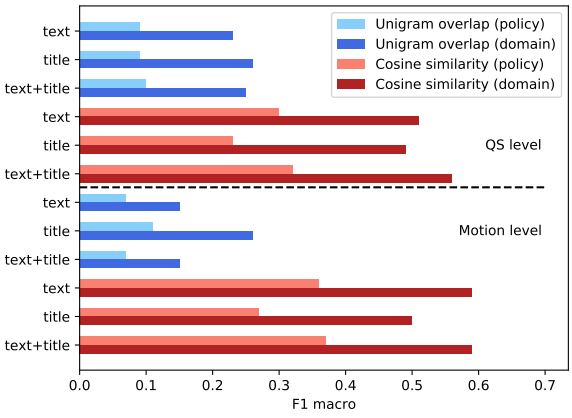


Figure 1: F1 macro scores for unigram overlap and cosine similarity matching at the policy and domain levels using textual features from whole motions. Use of cosine similarity leads to markedly better performance than unigram overlap, and the best performance is achieved using features derived from both the titles and motion texts at policy and domain levels.

Motions: Quasi-sentence Classification

We tested the supervised pipelines at the quasi-sentence level and at the two levels of class label granularity (policy and domain), which allows

Model	Text representation	Data Source	Policy		Domain	
			Macro	Micro	Macro	Micro
Unigram overlap	BOW	Motion titles	0.09	0.15	0.26	0.31
		Motions	0.09	0.21	0.23	0.38
		Titles+motions	0.10	0.23	0.25	0.39
Cosine similarity	Tf-idf	Motion titles	0.23	0.34	0.44	0.49
		Motions	0.30	0.36	0.50	0.51
		Titles+motions	0.32	0.41	0.51	0.56
SVM	Tf-idf	Motions	0.33	0.48	0.58	0.63
		Manifestos	0.29	0.40	0.53	0.56
	Domain w-emb	Motions	0.32	0.50	0.53	0.62
		Manifestos	0.25	0.41	0.45	0.53
	Wiki w-emb	Motions	0.35	0.51	0.55	0.65
		Manifestos	0.21	0.38	0.45	0.52
CNN	Domain w-emb	Motions	0.15	0.38	0.58	0.64
		Manifestos	0.19	0.30	0.37	0.51
	Wiki w-emb	Motions	0.13	0.29	0.50	0.57
		Manifestos	0.21	0.36	0.48	0.56
BERT	Large, cased	Motions	0.26	0.47	0.42	0.58
		Manifestos	0.32	0.47	0.52	0.57
		+ Motions fine-tuning	0.39	0.50	0.60	0.67
BERT+CNN	Large, cased	Motions	0.27	0.48	0.42	0.56
		Manifestos	0.29	0.44	0.54	0.60
		+ Motions fine-tuning	**0.47**	**0.57**	**0.61**	**0.69**

Table 6: F1 scores for similarity matching and classification of debate motions at the quasi-sentence level.

us to compare the results with previous work on the Manifesto Project (e.g., Zirn et al. (2016)). As can be seen in Table 6, the use of machine learning methods generally (but not always) leads to a substantial improvement (especially for Micro F1), in comparison to the heuristics that we have discussed above.

Concerning the SVM and CNN baselines, training the classifiers on the large collection of annotated manifestos and then applying them to the motions does not lead to improvements in comparison to the performance of the same architectures on the motions alone. Similarly, we notice that in most cases the use of in-domain embeddings does not improve the results. These two findings might be due to the fact that the style of communication and vocabulary of the employed resources are very different. The size of the training data may also play a role, as can be noticed in particular with the weak performances of the CNNs, especially in comparison to more traditional approaches; in the next section, we return to this issue.

Finally, to further confirm the large potential of BERT, even in tasks which involve many labels, a lack of training data, and a very specific style of communication, we have obtained a clear improvement over all other systems when employing this state-of-the-art architecture, trained on manifesto quasi-sentences and further fine-tuned on motions.

Manifestos: Quasi-sentence Classification

As a final comparison of the presented systems for quasi-sentence classification, we report their performance on the corpus of 16k manifesto quasi-sentences, again with an 80/20 train-test split. The results (see Table 7) are consistent with the performance of supervised pipelines on the Manifesto Corpus presented in previous literature (Glavaš et al., 2017a; Subramanian et al., 2018; Zirn et al., 2016) and in line with the performances we obtained on the motion corpus in Table 6.

Interestingly, we once again notice the weak performances of the CNNs on the collection, even with ten times as much training data. This could be due to a necessity to extend the architecture (for example, by adding more convolutional layers) rather than a simple lack of training data. Con-

Model	Text representation	Policy		Domain	
		Macro	Micro	Macro	Micro
SVM	Tf-idf	0.39	0.54	0.58	0.66
	Domain w-emb	0.35	0.53	0.52	0.64
	Wiki w-emb	0.38	0.54	0.54	0.66
CNN	Domain w-emb	0.28	0.47	0.54	0.58
	Wiki w-emb	0.27	0.44	0.52	0.56
BERT	Large, cased	**0.42**	**0.58**	0.58	0.64
BERT + CNN	Large, cased	**0.42**	**0.58**	**0.60**	**0.70**

Table 7: F1 scores for classification of party political manifestos at the quasi-sentence level.

versely, traditional SVM baselines offer reasonable results, and we achieve state-of-the-art performances when employing BERT.

8 Discussion and Conclusion

Through this work we have been able to make a number of observations about the validity and reliability of the annotations produced and the difficulty of the tasks of labelling both debate motions and manifestos.

In labelling the manifestos, our annotators agreed with each other to roughly the same extent that they agree with the gold labels provided by the Manifesto Project's expert annotators. This level of agreement is also similar to that reported in Mikhaylov et al. (2008), though not as good as that of MARPOR[12] itself (Lacewell and Werner, 2013).

The task does seem to be transferable to parliamentary debate motions, with our inter-annotator agreement scores comparable on both domains. Although automatic labelling with lexical similarity matching is more succesful at the quasi-sentence level than at the motion level, the annotators do not seem to find the coarser grained task much easier.

Overall, this is a hard task for humans. However, despite the issue of annotation reproducibility, political scientists continue to find these labels useful—as evidenced by Volkens et al. (2015), who find 230 articles that use this data in the eight journals they examine. With comparable reliabilty (inter-annotator agreement), the labelled motions could prove equally suitable for many automatic analysis applications.

Concerning automation of the labeling process, we can derive three general findings. The first is that a very simple approach—matching debate motions to coded manifestos using cosine similarity measurement—appears to produce potentially useful outputs, particularly at the domain level, with supervised baselines not necessarily offering consistently better results (especially the CNN architectures). The second is that cross-domain applications (from manifestos to motions) seem to necessitate a further fine-tuning step, perhaps due to the very different styles of communication involved. The third is the significant contribution that the use of BERT provides our supervised pipelines, which are able to achieve state-of-the-art performance on both the motions and manifesto quasi-sentences.

The generated dataset of topically labelled motions along with the trained BERT+CNN classifier can now pave the way for further work at the intersection of natural language processing and political science, which can benefit from these fine-grained policy position annotations: from analysing the sentiment of the motions to measuring the level of disagreement between members of the same party, and up to full-blown argumentation mining of each debate.

[12]Manifesto Research on Political Representation, the research team behind the Manifesto Project.

Acknowledgements

This work was supported in part by the SFB 884 on the Political Economy of Reforms at the University of Mannheim (projects B6 and C4), funded by the German Research Foundation (DFG). The authors would like to thank Melis Ince, Olga Sokolova, and Stefan Tasic for their diligent work on annotation, and the anonymous reviewers for their helpful comments.

References

Gavin Abercrombie and Riza Batista-Navarro. 2018a. 'Aye' or 'no'? Speech-level sentiment analysis of Hansard UK parliamentary debate transcripts. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Gavin Abercrombie and Riza Batista-Navarro. 2019. Sentiment and position-taking analysis of parliamentary debates: A systematic literature review. *arXiv preprint arXiv:1907.04126*.

Gavin Abercrombie and Riza Theresa Batista-Navarro. 2018b. Identifying opinion-topics and polarity of parliamentary debate motions. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 280–285, Brussels, Belgium. Association for Computational Linguistics.

Ian Budge, Hans-Dieter Klingemann, et al. 2001. *Mapping policy preferences: Estimates for parties, electors, and governments, 1945-1998*, volume 1. Oxford University Press.

Clint Burford, Steven Bird, and Timothy Baldwin. 2015. Collective document classification with implicit inter-document semantic relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 106–116.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kostas Gemenis. 2013. What to do (and not to do) with the Comparative Manifestos Project data. *Political Studies*, 61:3–23.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2016. Unsupervised text segmentation using semantic relatedness graphs. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics (*SEM)*.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2017a. Cross-lingual classification of topics in political texts. In *Proceedings of the Second Workshop on NLP and Computational Social Science (NLP+CSS)*, pages 42–46.

Goran Glavaš, Federico Nanni, and Simone Paolo Ponzetto. 2017b. Unsupervised cross-lingual scaling of political texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 688–693.

Alexander Herzog, Peter John, and Slava Jankin Mikhaylov. 2018. Transfer topic labeling with domain-specific knowledge base: An analysis of UK House of Commons speeches 1935-2014. *arXiv preprint arXiv:1806.00793*.

Werner Krause, Pola Lehmann, Jirka Lewandowski, Theres Matthie, Nicolas Merz, Sven Regel, and Annika Werner. 2018. Manifesto Corpus. version: 2018-1. Berlin: WZB Berlin Social Science Center.

Onawa P Lacewell and Annika Werner. 2013. Coder training: key to enhancing reliability and validity. *Mapping Policy Preferences from Texts*, 3:169–194.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Michael Laver, Kenneth Benoit, and John Garry. 2003. Extracting policy positions from political texts using words as data. *American Political Science Review*, 97(2):311–331.

Will Lowe, Kenneth Benoit, Slava Mikhaylov, and Michael Laver. 2011. Scaling policy preferences from coded political texts. *Legislative Studies Quarterly*, 36(1):123–155.

Stefano Menini, Federico Nanni, Simone Paolo Ponzetto, and Sara Tonelli. 2017. Topic-based agreement and disagreement in US electoral manifestos. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2938–2944.

Slava Mikhaylov, Michael Laver, and Kenneth Benoit. 2008. Coder reliability and misclassification in Comparative Manifesto Project codings. In *the 66th MPSA Annual National Conference*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Federico Nanni, Goran Glavaš, Simone Paolo Ponzetto, and Heiner Stuckenschmidt. 2019a. Political text scaling meets computational semantics. *arXiv preprint arXiv:1904.06217*.

Federico Nanni, Stefano Menini, Sara Tonelli, and Simone Paolo Ponzetto. 2019b. Semantifying the UK Hansard (1918-2018). In *Proceedings of the Joint Conference on Digital Libraries*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237.

Sven-Oliver Proksch and Jonathan B Slapin. 2010. Position taking in European Parliament speeches. *British Journal of Political Science*, 40(3):587–611.

Sven-Oliver Proksch and Jonathan B Slapin. 2015. *The politics of parliamentary debate*. Cambridge University Press.

James Pustejovsky and Amber Stubbs. 2012. *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*. O'Reilly Media, Inc.

Shivashankar Subramanian, Trevor Cohn, and Timothy Baldwin. 2018. Hierarchical structured model for fine-to-coarse manifesto text analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1964–1974.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 conference on Empirical Methods in Natural Language Processing*, pages 327–335. Association for Computational Linguistics.

Andrea Volkens, Cristina Ares, Radostina Bratanova, and Lea Kaftan. 2015. Scope, range, and extent of Manifesto Project data usage: A survey of publications in eight high-impact journals. In *Handbook for Data Users and Coders*. WZB.

Andrea Volkens, Judith Bara, Ian Budge, Michael D McDonald, and Hans-Dieter Klingemann. 2013. *Mapping policy preferences from texts: statistical solutions for manifesto analysts*, volume 3. OUP Oxford.

Annika Werner, Onawa Lacewell, and Andrea Volkens. 2011. Manifesto coding instructions: 4th fully revised edition.

Dani Yogatama, Lingpeng Kong, and Noah A. Smith. 2015. Bayesian optimization of text representations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2100–2105, Lisbon, Portugal. Association for Computational Linguistics.

Cäcilia Zirn, Goran Glavaš, Federico Nanni, Jason Eichorts, and Heiner Stuckenschmidt. 2016. Classifying topics and detecting topic shifts in political manifestos. In *Proceedings of the First International Conference on the Advances in Computational Analysis of Political Text (PolText)*.

Improving Neural Machine Translation by Achieving Knowledge Transfer with Sentence Alignment Learning

Xuewen Shi[1,2], **Heyan Huang**[1,2], **Wenguan Wang**[1,3], **Ping Jian**[1,2]*, **Yi-Kun Tang**[1,2]
[1]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[2]Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications
[3]Inception Institute of Artificial Intelligence, UAE
{xwshi, hhy63, wenguanwang, pjian, tangyk}@bit.edu.cn

Abstract

Neural Machine Translation (NMT) optimized by Maximum Likelihood Estimation (MLE) lacks the guarantee of translation adequacy. To alleviate this problem, we propose an NMT approach that heightens the adequacy in machine translation by transferring the semantic knowledge learned from bilingual sentence alignment. Specifically, we first design a discriminator that learns to estimate sentence aligning score over translation candidates, and then the learned semantic knowledge is transfered to the NMT model under an adversarial learning framework. We also propose a gated self-attention based encoder for sentence embedding. Furthermore, an N-pair training loss is introduced in our framework to aid the discriminator in better capturing lexical evidence in translation candidates. Experimental results show that our proposed method outperforms baseline NMT models on Chinese-to-English and English-to-German translation tasks. Further analysis also indicates the detailed semantic knowledge transfered from the discriminator to the NMT model.

1 Introduction

Recently, with the renaissance of deep learning, end-to-end Neural Machine Translation (NMT) (Kalchbrenner and Blunsom, 2013; Cho et al., 2014a; Sutskever et al., 2014; Bahdanau et al., 2014) has gained remarkable performance (Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017). Early NMT solutions are typically optimized to maximize the likelihood estimation (MLE) of each word in the ground truth translations during the training procedure. However, such an objective cannot guarantee the sufficiency of the generated translations in the NMT model, due to the lack of quantitative measurement for the information transformational completeness from the source side to the target side.

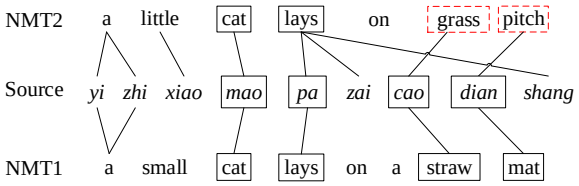


Figure 1: **Comparison between two Chinese-to-English translation examples of two independent NMT systems.** Lines between Source and NMTs represent model generated alignments (each source word cannot be covered more than once). Words in boxes are key words and red dotted dashed boxes indicate incorrect translations. Based on the model generated attention weights, NMT2 covers more source words than NMT1, which is opposite to human judgments.

Some existing work alleviates this problem by directly incorporating coverage or fertility mechanisms to an NMT model (Tu et al., 2016; Feng et al., 2016; Kong et al., 2019). However, the problem is that attention weights based coverage calculation for NMT is insensitive and sometimes even inaccurate to translation errors. Furthermore, it is unreasonable to consider the coverage of all kinds of source words equally, since various words contribute differently to sentences in semantics and syntax. For example, as illustrated in Fig. 1, translation errors are recorded as positive examples, and the alignments between function words also dilute the impact of key words alignments.

In this paper, we address the problem of inadequate translation by introducing a novel sentence alignment constrain under an adversarial training framework (Goodfellow et al., 2014; Lu et al., 2017; Yang et al., 2018). Specifically, our approach contains two sub-models: i) a sentence alignment oriented discriminator D learns to estimate the alignment score and sort the translation candidates by mainly considering the weighted

*Corresponding author: Ping Jian

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 260–270
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

alignment pairs (Ma, 2006) at a sentence representation level; and ii) a standard NMT model G aims to produce an appropriate translation with the highest ranking score (assigned by D) in the candidate list. To better capture the semantic alignment evidence of the input data, we also propose a novel gated self-attention based encoder for bilingual sentences encoding in discriminator D. Then, an N-pair training loss (Sohn, 2016) is introduced to select appropriate translation results from the candidates. We also leverage Gumbel-Softmax (GS) (Jang et al., 2017; Kusner and Hernández-Lobato, 2016) approximation for G to solve the problem of discrete samples, making the response from D to G differentiable.

To sum up, the proposed approach has the following advantages:

- We apply a novel end-to-end NMT adversarial training framework that heightens adequacy in translation. Under the framework, an NMT model is encouraged to generate translations that match semantic knowledge learned by a discriminator for sentence aligning, which can be viewed as an instance of "knowledge transfer".

- Benefited from the gated self-attention mechanism, the proposed encoder learns to focus on important lexical evidence for sentence aligning and enhance the contribution of the key words. This knowledge will be transfered to G through the proposed framework.

- The N-pair loss (Sohn, 2016; Lu et al., 2017) encourages samples closed to the gold-standard one to get higher score. Unlike a binary classification used in previous work (Yu et al., 2017; Yang et al., 2018; Wu et al., 2018), translations that are correct but different from the ground-truth ones will not be over penalized.

We use one of the state-of-the-art NMT models, Transformer (Vaswani et al., 2017), as the baseline model architecture and conduct experiments on Chinese-to-English and German-to-English translation tasks. Experimental results show that our proposed approach achieves significant improvements on both language pairs. We also evaluate the performance of the discriminator on both sentence alignment and translation candidate reranking tasks, which proves its independence and transferability. Further analyses show the specific alignment-oriented knowledge that the discriminator transfers to the NMT model.

2 Related Work

Most of the state-of-the-art NMT models are optimized by MLE-based objectives (Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017), but likelihood fails to measure whether the source information is completely transformed to the target side. Thus, it cannot handle translation adequacy problem (Tu et al., 2017).

One way to alleviate these problems is to apply coverage and fertility to NMT model. Feng et al. (2016) aim at controlling the fertilities of source words by appending additional additive terms to train objectives. Tu et al. (2016) employ coverage vector or coverage ratio as a lexical-level indicator to represent whether a source word is translated or not.

On the other hand, some recent efforts introduce additional source side constraints and explore duality properties of NMT (He et al., 2016; Cheng et al., 2016; Xia et al., 2017; Tu et al., 2017). Cheng et al. (2016) present a semi-supervised approach to train bidirectional NMT models and reconstruct the monolingual corpora using an auto-encoder (Socher et al., 2011). Tu et al. (2017) add a re-constructor to traditional NMT model, which introduces an auxiliary score to measure the adequacy of translation. Dual learning (He et al., 2016) and dual supervised learning (Xia et al., 2017) are also proposed to exploit the probabilistic correlation between dual tasks to regularize the training process. These previous approaches apply a reconstruction reward by comparing the source input and the reconstructed sentence, while we use alignment score directly to model the discrepancy between the source and the translation.

GAN (Goodfellow et al., 2014) is another promising framework to leverage sentence-level objectives in NMT. Recently, there is some remarkable work in NMT (Wu et al., 2018; Yang et al., 2018). The framework comprises of two sub-models: i) an NMT model aims to produce sentences which are hard to be discriminated from the gold-standard sentences; and ii) a discriminator makes efforts to differentiate the model generated translations from the ground-truth ones. A policy gradient method is leveraged to co-train the NMT model and the discriminator. However,

those approaches rarely take account of translation adequacy. Furthermore, the discriminators of those work refer the target sentence in the corpus as the single gold-standard regardless the quality of model generated translations, which will punish too much to the good model generated translations. Kong et al. (2019) propose an adequacy-oriented discriminator which is trained to estimate the Coverage Difference Ratio (CDR) given the source and the generated translation. However, CDR is unable to distinguish translation errors and it also neglects the importance of diversity between different words (as the examples shown in Fig. 1).

Unlike the discriminators in (Wu et al., 2018; Yang et al., 2018; Kong et al., 2019), our alignment-oriented discriminator learns a specific function to measure alignment score between source and target sentences, which is trained totally independently by the NMT generator. The proposed discriminator assigns different weights to words and is sensitive to translation errors. We also apply N-pair loss for training D to ensure that D will not punish the translations closed to the gold-standard overly.

3 Approach

In this section, we describe our approach that can transfer knowledge from a sentence alignment oriented discriminator D to an NMT model G. Our approach mainly consists of two sub-models: i) a discriminator D learns to estimate the alignment score and sort the translation candidates, and ii) an NMT model G aims to generate translations with higher score assigned by D. A sketch of the proposed training framework is shown in Fig. 2: for each sentence pair (X, Y) sampled from the training corpus, the NMT model G generates a translation $\hat{\mathrm{Y}}$ given X, and queries the discriminator D with $\hat{\mathrm{Y}}$ to get feedback and update itself. In order to obtain more stable training, we also leverage a teacher-forcing (Li et al., 2017) step to our approach.

3.1 NMT Generator

In this paper, we take the Transformer (Vaswani et al., 2017), one of the popular state-of-the-art NMT models, as the specific implementation of the NMT model G. This helps to better illustrate the effectiveness of the proposed method. The Transformer in this paper follows the conventional encoder-decoder framework (Cho et al., 2014b). Specifically, the encoder contains a stack of six identical layers. Each layer is consist of two sub-layers: i) a multi-head self-attention mechanism, and ii) a position-wise fully connected feed-forward network. A residual connection is applied around each of the two sub-layers, followed by layer normalization (Ba et al., 2016). The decoder is also composed of a stack of six identical layers. Besides the two sub-layers stated above, a third sub-layer is inserted in each layer that performs multi-head attention over the output of the encoder.

Following the base model setups of the Transformer (Vaswani et al., 2017), we use 8 attention heads, 512-dimensional output vectors for each layer, and 2048-dimensional inner-layer of the feed-forward network.

1: Pre-train a generator G (see section 3.1) and a discriminator D (see section 3.2), individually.
2: **for** number of training iterations **do**
3: Sample $(\mathrm{X}, \mathrm{Y}^{+})$ from training corpus
4: Sample $\hat{\mathrm{Y}} \sim G(\mathrm{X})$ with a Gumbel-Softmax sampler (see section 3.4)
5: Compute loss $\mathcal{L}_G$ for $(\mathrm{X}, \hat{\mathrm{Y}})$ using D (see section 3.3)
6: Update G with the learning rate η:
$$\theta_G \leftarrow \theta_G - \eta \bigtriangledown_{\theta_G} \mathcal{L}_G \quad (1)$$
7: Teacher-Forcing: update G on $(\mathrm{X}, \mathrm{Y}^{+})$ (see section 3.5)
8: **end for**

Figure 2: A brief overview of the proposed training framework. See section 3 for more details.

3.2 Discriminator

For the discriminator D, we propose a gated self-attention based sentence encoder to perform source and target sentence encoding, and then calculate the alignment score using the encodings pair.

Gated Self-Attention Sentence Encoder. As depicted in Fig. 3, we opt a shallow network architecture: one gated hidden layer and one self-attention layer as the sentence encoder. This lightweight encoder mainly captures the lexical meanings of the sentence. The self-attention mechanism helps the encoder select more impor-

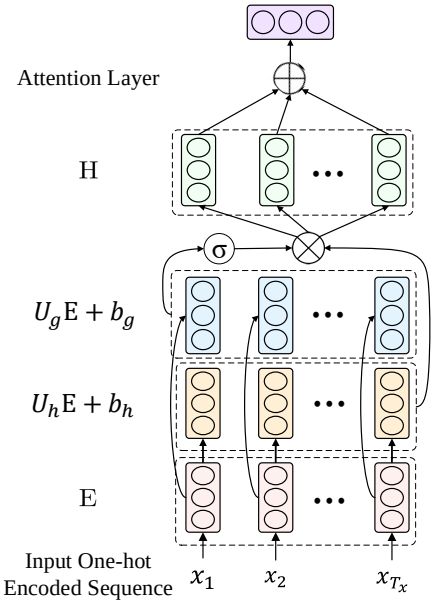


Figure 3: **Model architecture of the gated self-attention sentence encoder.** See section 3.2 for more details.

tant lexical evidences to estimate the alignment score between two sentences.

Given a one-hot encoded input sequence X and a word embedding lookup table $\mathrm{D} \in \mathbb{R}^{|V|\times d}$, where d is the model dimension. The input X will be represented as a corresponding word embedding matrix $\mathrm{E} \in \mathbb{R}^{\mathrm{T_x}\times \mathrm{d}}$. We apply a gating mechanism (Dauphin et al., 2017) to compute the hidden layer H:

$$\mathrm{H} = (U_h\mathrm{E} + b_h) \otimes \sigma(U_g\mathrm{E} + b_g), \tag{2}$$

where U_h and $U_g \in \mathbb{R}^{d\times d}$, $\sigma(\cdot)$ is a logistic sigmoid function and $\otimes$ is element-wise product between matrices. Then the self-attention weights W is computed as:

$$\mathrm{W} = \mathrm{softmax}(\tanh(U_a\mathrm{H})), \tag{3}$$

where $U_a \in \mathbb{R}^{d\times d}$. The output of the gated self-attention encoder is formulated as:

$$e = U_o(\mathrm{W} \times \mathrm{H}) + b_o, \tag{4}$$

where $e \in \mathbb{R}^d$, and $U_o \in \mathbb{R}^{d\times d}$. We add layer normalization (Ba et al., 2016) to the output layer. The model dimension d is set to 512.

Alignment Score and Discriminator Loss. With the source and target sentence encodings e_x and e_y, the alignment score $s_{(\mathrm{X},\mathrm{Y})}$ can be computed as:

$$s_{(\mathrm{X},\mathrm{Y})} = e_x^\top e_y. \tag{5}$$

Given the candidate target sentences list $\mathcal{Y}$, the discriminator produces a distribution over $\mathcal{Y}$ and aims to maximize the log-likelihood of the gold-standard alignment sentence Y^+. Since sentence-level alignments in automatic extracted corpora are usually not very precise, we expect the loss function for training D not to be too strict with candidates that are closed to the gold-standard one. Therefore, following Lu et al. (2017), we apply a metric-learning multi-class N-pair loss (Sohn, 2016) to our model, which can be defined as:

$$\begin{aligned}\mathcal{L}_D &= \mathcal{L}_{N-\mathrm{pair}}(\{\mathrm{X}, \mathrm{Y}^+, \{\mathrm{Y}_n^-\}_{n=1}^{N-1}\}) \\ &= \log(1 + \sum_{n=1}^{N} \exp(s_{(\mathrm{X},\mathrm{Y}_\mathrm{n}^-)} - s_{(\mathrm{X},\mathrm{Y}^+)})),\end{aligned} \tag{6}$$

where Y^+ is alignment target sentence to the source language X, and Y_n^- is one of the $N-1$ unaligned samples.

Compared to cross entropy loss used in previous work (Yu et al., 2017; Yang et al., 2018; Wu et al., 2018), the N-pair objective encourages the score of target sentences which are similar to the given golden-standard one to be higher than the dissimilar ones. In this way, translations that are correct but different from the ground truth will not be over penalized, and thus this can be useful to provide a reliable signal for the generator.

In later sections, we will analyze the semantic information learned by the model through some visualization examples, shown in section 5.1, and the experimental results show that it achieves sufficient accuracy for scoring the alignment between source and target sentences.

3.3 Discriminative Losses for Generative Training

In our framework, G aims to generate a translation score higher than the golden-standard, under the premise of encoders and the scoring function learned by D. Specifically, for each sentence pairs $(\mathrm{X}, \mathrm{Y}^+)$ in training sets, first, G samples translation $\hat{\mathrm{Y}}$ given X with greedy searching. Second, D takes $\hat{\mathrm{Y}}$ as well as $(\mathrm{X}, \mathrm{Y}^+)$ as inputs to compute alignment scores, and then G gets the feedback from D. Eq. (7) gives the perceptual loss that G aims to optimize.

$$\begin{aligned}\mathcal{L}_G &= \mathcal{L}_{1-pair}(\{\mathrm{X}, \mathrm{Y}^+, \hat{\mathrm{Y}}\}) \\ &= \log(1 + \exp(s_{(\mathrm{X},\mathrm{Y}^+)} - s_{(\mathrm{X},\hat{\mathrm{Y}})})).\end{aligned} \tag{7}$$

Intuitively, updating generator parameters to minimize $\mathcal{L}_G$ can be interpreted as learning to produce a translation $\hat{Y}$ that "fools" the discriminator into believing that this answer should score higher than the human response Y^+ under the D's scoring function.

3.4 Gumbel-Softmax Sampler

The process of sampling a translation $\hat{\mathrm{Y}}$ with G is not differentiable, since it includes $\mathrm{argmax}(\cdot)$ operator to perform one-hot encoding. We leverage the Gumbel-softmax (Jang et al., 2017) sampler to solve this problem. Formally, at the decoding step j, suppose that $p_j \in \mathbb{R}^{V_y}$ contains the model output log-probabilities over target vocabulary, and $g_j \in \mathbb{R}^{V_y}$ includes i.i.d samples drawn from the standard distribution $\mathrm{Gumbel}(0, 1)$. A sample y_j is transformed as:

$$\hat{y}_j = \mathrm{softmax}((p_j + g_j)/\tau), \tag{8}$$

where τ is a temperature parameter and is set to 0.5 in our experiments.

3.5 Teacher-forcing Step

Since $\mathcal{L}_G$ in Eq. (7) mainly considers the discrepancy of alignment and integrity between the model output and the ground-truth, it rarely inspects grammar correctness and language fluency. To alleviate this problem, following (Li et al., 2017; Lu et al., 2017), we adopt the similar teacher-forcing step to our training process.

We perform two different teacher-forcing objectives for comparison: i) a likelihood objective O_{LM} and ii) a BLEU score reward ($\mathrm{R}_{\mathrm{BLEU}}$), under the training strategies of MLE and MRT (Shen et al., 2016), respectively.

4 Experiments

4.1 Datasets and Setups[1]

We evaluate the proposed approach on Chinese-to-English (Zh-En) and English-to-German (En-De) translation tasks. For both of the two translation tasks, we tokenize all corpora with the Moses tokenizer[2]. Sentences longer than 100 words are discarded, and all the sentences are encoded with byte-pair encoding (bpe) (Sennrich et al., 2016).

Chinese-to-English. For Chinese-to-English translation, our training data are extracted from four LDC corpora[3]. The training set contains totally 1.3M parallel sentence pairs. For preprocessing, the Chinese part for both training and testing sets is segmented by the LTP Chinese word segmentor (Che et al., 2010) before applying bpe (Sennrich et al., 2016) to the corpus. We get a Chinese vocabulary of about 39K tokens, and an English vocabulary of about 30K tokens. We use NIST2005 dataset for validation and NIST2002, NIST2003, and NIST2004 datasets for testing. In the following parts of the paper, the Chinese examples are presented by segmented italic romanized form, and different Chinese characters are delimited by single quotation marks.

English-to-German. For English-to-German translation, we conduct experiments on the publicly available corpora WMT'14 En-De. The training set of En-De task totally contains 4.5M sentence pairs, and we use a shared source-target vocabulary of about 39K tokens. We use newstest2013 as the validation set and report the results on newstest2014.

Discriminative Corpus Construction. Different from parallel sentence pairs for training generative models, the corpus for training D needs to provide a candidate translations list for each source sentence. Therefore, we need to manually construct the corpus for training D using the original parallel corpus. For each source sentence, we set the size of candidate list to 100. The translation candidates are preferentially obtained from the context of the golden standard translation in the comparable paragraph. If the context sentence number N_c is less than 99, we will randomly sample another $99 - N_c$ sentences from the whole rest target corpus. As for the data format, we follow most of Das et al. (2019).

Evaluation. As for generative models, following Vaswani et al. (2017), we report the result of a single model obtained by averaging the 5 checkpoints around the best model selected on the development set. We apply beam search during decoding with the beam size of 6. The translation results in this paper are measured in case-insensitive BLEU (Papineni et al., 2002) by the

[1]The demo data and source codes will be released online at https://github.com/PolarLion/Sentence-Alignment-Learning

[2]https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

[3]LDC2005T10, LDC2003E14, LDC2004T08 and LDC2002E18. Since LDC2003E14 is a document-level alignment comparable corpus, we use Champollion Tool Kit (Ma, 2006) to extract parallel sentence pairs from it.

multi-bleu.perl script[4]. For the discriminator, the performance is evaluated on recall@k and mean rank score.

4.2 Training Details

Pre-train discriminator and NMT model. During the training procedure, we first pre-train the discriminator and the generator separately for a warm start. We pre-train the model until the performance of D and G on development set dose not improve. For training discriminator on all language pairs, we use the Adam optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.99$ and a base learning rate of 4×10^{-4}. The mini-batch size is 100 and the dropout rate is set to 0.1. As for the NMT model, we follow the base model of Transformer (Vaswani et al., 2017) for most of training setups, except the label smoothing (Szegedy et al., 2016).

Frozen discriminator v.s. adversarial discriminator. We also study the effects of two training setups of the discriminator: updating D (adversarial D) or not (frozen D) with the NMT model. When we perform frozen D, the NMT model is learned with a combination discriminative perceptual loss (Lu et al., 2017) and teacher-forcing loss (Li et al., 2017; Lu et al., 2017). Each mini-batch contains 30 sentence pairs due to the limitation of memory size of a single GPU. For the adversarial discriminator, We alternately update G and D under the adversarial learning framework (Yang et al., 2018; Wu et al., 2018; Kong et al., 2019). An adversarial D is to maximize the score of the human translation Y^+ and minimize the score of the generated translation $\hat{\mathrm{Y}}$. Then the training loss for adversarial D can be represented as: $\mathcal{L}_D = -\mathcal{L}_G$.

4.3 Machine Translation Results

We report the experimental results on machine translation in this section. Table 1 shows the BLEU scores of Zh-En and En-De translation tasks. Our approach achieves an improvement up to +0.76 BLEU points averagely on Zh-En testing sets and +0.64 BLEU points on En-De testing set. It should be noted that we do not apply label smoothing (Szegedy et al., 2016) due to using Gumbel-Softmax approximation, which results in a decline in En-De translation performance compared to the reported result of BLEU 27.3 in Vaswani et al. (2017).

We compare two setups of frozen D (Row 3-4) and adversarial D (Row 5-6) for the discriminator. Experimental results show that continuing to update D along with G gains best BLEU score for the both translation tasks. It means that fine-tuning D with the model generated data can further improve training quality. We also evaluate two different training objectives for the teacher-forcing step (Row 3,5 vs Row 4,6) and Row 2 is another baseline for training only on $\mathrm{R}_{\mathrm{BLEU}}$ (similar to MRT (Shen et al., 2016)). We can see from the results that applying MLE or BLEU reward does not make much difference. These results indicate that the proposed method makes up for the shortcomings of MLE training.

4.4 Discriminative Results

The format of the test datasets for discriminator is similar to the training set described in section 4.1, where each input corresponds to one hundred candidate translations extracted from the document context. The goal of the discriminator is to rank the correct translation as high as possible. We present recall@k and mean rank of the discriminator on Zh-En and En-De test sets in Table 2. It shows that for all test sets of both language pairs, our proposed discriminator performs steadily at high recall rate of more than 96% on recall@1 and nearly 100% on recall@5 and recall@10. Both of the high recall@k and ranking mean closed to 1 indicate that the ground-truth translations are always assigned to high alignment score.

Empirical and principled studies indicate that high initial accuracy of binary classification based discriminator may lead to worse model performance for GANs (Salimans et al., 2016; Arjovsky and Bottou, 2017; Yang et al., 2018). In this paper, G is trained with a specific 1-pair loss defined on sentence alignment score, instead of Jensen-Shannon divergence (Arjovsky and Bottou, 2017; Arjovsky et al., 2017) between two data distributions, which could avoid the vanishing gradient problem in GANs. Therefore, the high accuracy of the proposed discriminator would not make negative impact on G.

5 Analysis

In this section, we will study characteristics of the proposed approach and report some detailed ex-

[4] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

#	Model	Zh-En				En-De
		NIST2002	NIST2003	NIST2004	Average	newstest2014
1	Transformer	41.56	39.95	42.05	41.19	26.52
2	+$\mathrm{R_{BLEU}}$	42.30	40.47	42.46	41.74	26.73
3	+ frozen D + $\mathrm{O_{LM}}$	42.51	**40.74**	42.42	41.89	27.10
4	+ frozen D + $\mathrm{R_{BLEU}}$	41.63	40.06	42.16	41.28	26.77
5	+ adversarial D +$\mathrm{O_{LM}}$	42.67	40.67	42.51	**41.95**	**27.16**
6	+ adversarial D +$\mathrm{R_{BLEU}}$	**42.75**	40.28	**42.67**	41.90	27.05

Table 1: **BLEU scores on Zh-En and En-De translation task.** Transformer is the baseline model. "Average" is the averaged BLEU scores on testing sets. Following Vaswani et al. (2017), we report the result of a single model obtained by averaging the 5 checkpoints around the best model selected on the development set. See section 4.3 for more details.

Testset	R@1	R@5	R@10	Mean
NIST2002	97.04	99.77	99.89	1.06
NIST2003	97.50	99.34	99.67	1.10
NIST2004	97.25	99.94	99.94	1.05
newstest2014	96.60	99.43	99.73	1.12

Table 2: **Discriminator performance on Zh-En and En-De test sets.** R@k and Mean are abbreviations for recall@k score and mean rank score, respectively. See section 4.4 for more details.

Model setups	Align
Transformer	11.15
+$\mathrm{R_{BLEU}}$	11.18
+ frozen D + $\mathrm{O_{LM}}$	11.31
+ frozen D + $\mathrm{R_{BLEU}}$	**11.38**
+ adversarial D +$\mathrm{O_{LM}}$	11.34
+ adversarial D +$\mathrm{R_{BLEU}}$	11.36

Table 3: **Averaged sentence alignment scores on Zh-En NIST2002∼2004 test sets.** "Align" means the averaged sentence alignment score estimated by D. The higher score represents the better alignment quality in D's view. See section 5.1 for more details.

perimental results. We also give a specific translation example to illustrate how knowledge transferring improves NMT performance.

5.1 What kind of Knowledge Does D Transfer to G?

In this paper, we propose a discriminator that directly learns to measure alignment and then transfers the learned knowledge to an NMT model. D is designed to capture lexical evidence for sentence alignment by learning a self-attention encoder. We give averaged sentence alignment scores between translations and source inputs on different model setups in Table 3. Those alignment scores are estimated by a pre-trained D. Table 3 shows that the output alignment scores of the proposed approaches are all higher than the baseline methods, which illustrates that G can learn the knowledge on measuring alignment from D under the proposed training frameworks.

In order to illustrate the lexical-level knowledge learned by D, we give a visual example in Fig. 4. It shows self-attention weights of the encoders for the given source and the target sentence. In the example, the source sentence is "*bao'wei'er 12'ri yu sha'long ju'xing le hui'tan*" and the target sentence is "Powell hold a talk with Sharon on the 12th ." We notice that the source language words "*bao'wei'er*", "*12'ri*" and "*sha'long*", and their corresponding target language words "Powell", "12th" and "Sharon" are assigned higher attention weights than others. This means that the encoders regard those words as important lexical evidences for estimating the alignment score. Those self-learned attention weights share the same spirit with the weighted translation pairs in Champollion (Ma, 2006). During the training process, G leans to treat those important words carefully and avoid missing them to get higher score with the judgment of D. This process can be considered as transferring the semantic knowledge leaned by D to G.

5.2 Can discriminator distinguish good and bad translation results?

Since D is trained independently in our framework, it is difficult to estimate whether the discriminator can correctly distinguish the good and bad translations generated by the NMT model. Therefore, to verify whether an individual discriminator is suitable for the model generated data,

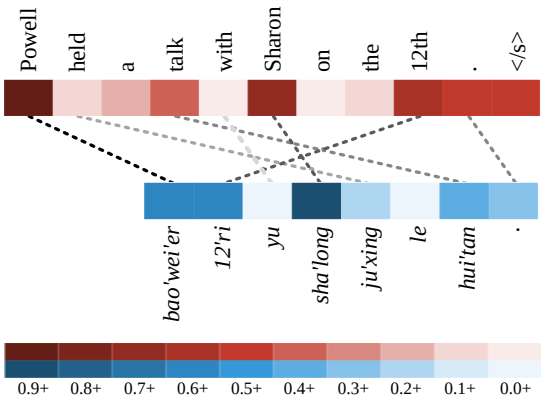


Figure 4: **Example of the self-attention weights for the source (lower) and the target (upper) language encoders.** Sentences in the example are selected from NIST2002 Zh-En test set. All weights in this example are scaled by Min-Max scaling method for better visualization and darker colors represent higher attention weights. Aligned words are manually connected by dashed lines. See section 5.1 for more details.

we conduct further experiments on translation re-ranking task on the baseline Transformer (Vaswani et al., 2017) model on Zh-En translation. Moreover, in order to obtain more translation candidates, we expand the beam size to 24 and then re-order the N-best translation candidates by D. Experimental results are shown in Table 4. We can observe that the larger beam search size leads to the worse performance, since the likelihood score for decoding tends to score short translations higher than long sentences. Larger searching space also brings more good translation candidates, and D re-orders them by alignment score and gains better BLEU scores than most baseline setups as shown in Table 4. The above observation indicates that D can successfully handle the unseen data generated by NMT models. Previous work (Wu et al., 2016; Koehn and Knowles, 2017) introduces length normalization to solve the above beam search decoding problem, whose results are also presented in Table 4 for a fair comparison.

5.3 Example Translations

We provide example translations on Zh-En translation task in Fig. 5. From Fig. 5, we can see that though the translation results of the baseline model is correct in syntax, its logic is wrong on account of missing an important source information of "went to Hong Kong on Saturday for a visa". All the translations generated by our proposed method do not make this mistake, since it is learned and

Setups	NIST02	NIST03	NIST04
beam 6	41.56	39.95	42.05
beam 24	40.72	38.64	41.13
+length penalty	41.93	**40.26**	42.49
+re-ranking	**42.22**	40.20	**42.56**

Table 4: **BLEU scores on Zh-En translation re-ranking task.** The "beam N" represents the decoding beam search size. The "+length penalty" means using length normalization (Wu et al., 2016) when performing beam search. The "+re-ranking" represents that the translation candidates are re-ranked by D. See section 5.2 for more details.

transfered from the discriminator where the verb "went", nouns 'Saturday' and "visa" are important lexical evidence for estimating alignment score. We also show a translation re-ranking example, which gains a similar result to other proposed methods. An alignment score evaluated by the discriminator and a sentence-level BLEU[5] (Papineni et al., 2002) score are also shown under the corresponding translations. Both the golden reference and the model generated translations gain higher alignment score from D, which illustrates the rationality of discriminator design.

6 Conclusion

In this work, we propose a novel training framework which achieves sentence alignment oriented knowledge transfer to improve the NMT. We design a discriminator to measure sentence alignment by mainly considering lexical evidence via a gated self-attention mechanism. Then, a discriminative loss as well as a teacher-forcing objective is used to make NMT model generate sufficient and fluent translations during training procedure. Experimental results on different language pairs show that our proposed approach outperforms standard NMT models. Further analysis indicates the proposed discriminator well captures the weighted lexical relationships among sentences and successfully transfers the knowledge to the NMT model.

In the future, we would like to make discriminator learn more semantic related knowledge like dependency, and combine our approach with other advanced techniques in reinforcement learning and adversarial learning (Yu et al., 2017; Yang et al., 2018; Kong et al., 2019).

[5]Evaluated by Moses (Koehn et al., 2007) sentence-bleu script.

Source	*chen'jin'de* ***xing'qi'liu fu*** *xiang'gang* ***qu'de qian'zheng*** *, zuo'tian di jing fang'wen 10 tian .*
Reference	Chen Chin-teh **went to** Hong Kong **on Saturday for his visa** and arrived in Beijing yesterday for his 10-day visit . *(align: 11.15)*
Transformer	Chen Jinde arrived in Beijing yesterday for a 10-day visit to Hong Kong . *(align: 9.91, BLEU: 28.33)*
+frozen D re-ranking	after **receiving a visa** in Hong Kong **on Saturday** , Chen Jinde arrived in Beijing yesterday for a 10-day visit . *(align: 10.75, BLEU: 39.32)*
+frozen D +O_{LM}	Chen Jinde **went to** Hong Kong **to obtain a visa on Saturday** and yesterday arrived in Beijing for a 10-day visit . *(align: 10.97, BLEU: 29.55)*
+frozen D +R_{BLEU}	Chen Jinde **went to** Hong Kong **on Saturday to obtain a visa** , and yesterday arrived in Beijing for a 10-day visit . *(align: 10.99, BLEU: 37.49)*
+adversarial D +O_{LM}	Chen Jinde **went to** Hong Kong **on Saturday to obtain a visa** and yesterday arrived in Beijing for a 10-day visit . *(align: 10.92, BLEU: 40.19)*
+adversarial D +R_{BLEU}	Chen Jinde **went to** Hong Kong **on Saturday for a visa** , and yesterday arrived in Beijing for a 10-day visit . *(align: 11.01, BLEU: 44.53)*

Figure 5: **Example translations on the Zh-En translation task.** The example is selected from the NIST2002 testing set. "Source" and "Reference" are the source input and one of the four given references. **Words in red bold fonts** represent the missing part of the translation generated by the baseline model. A alignment score (*align*) and a sentence-level BLEU are given below the target sentence. See section 5.3 for more details.

Acknowledgments

We thank all anonymous reviewers for their valuable comments. This work was supported by the National Key Research and Development Program of China (Grant No. 2017YFB1002103) and the National Natural Science Foundation of China (No. 61732005).

References

Martín Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. *CoRR*, abs/1701.07875.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. LTP: A chinese language technology platform. In *COLING 2010, 23rd International Conference on Computational Linguistics, Demonstrations Volume, 23-27 August 2010, Beijing, China*, pages 13–16.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Semi-supervised learning for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, Stefan Lee, José M. F. Moura, Devi Parikh, and Dhruv Batra. 2019. Visual dialog. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(5):1242–1256.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 933–941.

Shi Feng, Shujie Liu, Nan Yang, Mu Li, Ming Zhou, and Kenny Q. Zhu. 2016. Improving attention modeling with implicit distortion and fertility for machine translation. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 3082–3092.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on*

Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, pages 1243–1252.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial networks. *CoRR*, abs/1406.2661.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 820–828.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1700–1709.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation, NMT@ACL 2017, Vancouver, Canada, August 4, 2017*, pages 28–39.

Xiang Kong, Zhaopeng Tu, Shuming Shi, Eduard H. Hovy, and Tong Zhang. 2019. Neural machine translation with adequacy-oriented learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, pages 6618–6625.

Matt J. Kusner and José Miguel Hernández-Lobato. 2016. GANS for sequences of discrete elements with the gumbel-softmax distribution. *CoRR*, abs/1611.04051.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2157–2169.

Jiasen Lu, Anitha Kannan, Jianwei Yang, Devi Parikh, and Dhruv Batra. 2017. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 313–323.

Xiaoyi Ma. 2006. Champollion: A robust parallel text sentence aligner. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006, Genoa, Italy, May 22-28, 2006.*, pages 489–492.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2226–2234.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 151–161.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing*

Systems 2016, December 5-10, 2016, Barcelona, Spain, pages 1849–1857.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826.

Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2017. Neural machine translation with reconstruction. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3097–3103.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Lijun Wu, Yingce Xia, Fei Tian, Li Zhao, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. Adversarial neural machine translation. In *Proceedings of The 10th Asian Conference on Machine Learning, ACML 2018, Beijing, China, November 14-16, 2018.*, pages 534–549.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Yingce Xia, Tao Qin, Wei Chen, Jiang Bian, Nenghai Yu, and Tie-Yan Liu. 2017. Dual supervised learning. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3789–3798.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Improving neural machine translation with conditional sequence generative adversarial nets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1346–1355.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 2852–2858.

Code-Switched Language Models Using Neural Based Synthetic Data from Parallel Sentences

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, Pascale Fung
Center for Artificial Intelligence Research (CAiRE)
Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
{giwinata,amadotto,cwuak}@connect.ust.hk, pascale@ece.ust.hk

Abstract

Training code-switched language models is difficult due to lack of data and complexity in the grammatical structure. Linguistic constraint theories have been used for decades to generate artificial code-switching sentences to cope with this issue. However, this require external word alignments or constituency parsers that create erroneous results on distant languages. We propose a sequence-to-sequence model using a copy mechanism to generate code-switching data by leveraging parallel monolingual translations from a limited source of code-switching data. The model learns how to combine words from parallel sentences and identifies when to switch one language to the other. Moreover, it captures code-switching constraints by attending and aligning the words in inputs, without requiring any external knowledge. Based on experimental results, the language model trained with the generated sentences achieves state-of-the-art performance and improves end-to-end automatic speech recognition.

1 Introduction

Code-switching is a common linguistic phenomenon in multilingual communities, in which a person begins speaking or writing in one language and then switches to another in the same sentence.[1] It is motivated in response to social factors as a way of communicating in a multicultural society. In its practice, code-switching varies due to the traditions, beliefs, and normative values in the respective communities. Linguists have studied the code-switching phenomenon and proposed a number of linguistic theories (Poplack, 1978; Pfaff, 1979; Poplack, 1980; Belazi et al., 1994). Code-switching is not produced indiscriminately, but follows syntactic constraints. Many linguists have formulated various constraints to define a general rule for code-switching (Poplack, 1978, 1980; Belazi et al., 1994). However, these constraints cannot be postulated as a universal rule for all code-switching scenarios, especially for languages that are syntactically divergent (Berk-Seligson, 1986), such as English and Mandarin since they have word alignments with an inverted order.

Building a language model (LM) and an automatic speech recognition (ASR) system that can handle intra-sentential code-switching is known to be a difficult research challenge. The main reason lies in the unpredictability of code-switching points in an utterance and data scarcity. Creating a large-scale code-switching dataset is also very expensive. Therefore, code-switching data generation methods to augment existing datasets are a useful workaround.

Existing methods that apply equivalence constraint theory to generate code-switching sentences (Li and Fung, 2012; Pratapa et al., 2018) may suffer performance issues as they receive erroneous results from the word aligner and the part-of-speech (POS) tagger. Thus, this approach is not reliable and effective. Recently, Garg et al. (2018) proposed a SeqGAN-based model to generate code-switching sentences. Indeed, the model learns how to generate new synthetic sentences. However, the distribution of the generated sentences is very different from real code-switching data, which leads to underperforming results.

To overcome the challenges in the existing works, we introduce a neural-based code-switching data generator model using pointer-generator networks (Pointer-Gen) (See et al., 2017) to learn code-switching constraints from a limited source of code-switching data and leverage their translations in both languages. Intu-

[1] Code-switching refers to mixing of languages following the definitions in Poplack (1980). We use "intra-sentential code-switching" interchangeably with "code-mixing".

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 271–280
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

itively, the copy mechanism can be formulated as an end-to-end solution to copy words from parallel monolingual sentences by aligning and reordering the word positions to form a grammatical code-switching sentence. This method solves the two issues in the existing works by removing the dependence on the aligner or tagger, and generating new sentences with a similar distribution to the original dataset. Interestingly, this method can learn the alignment effectively without a word aligner or tagger. As an additional advantage, we demonstrate its interpretability by showing the attention weights learned by the model that represent the code-switching constraints. Our contributions are summarized as follows:

- We propose a language-agnostic method to generate code-switching sentences using a pointer-generator network (See et al., 2017) that learns when to switch and copy words from parallel sentences, without using external word alignments or constituency parsers. By using the generated data in the language model training, we achieve the state-of-the-art performance in perplexity and also improve the end-to-end ASR on an English-Mandarin code-switching dataset.

- We present an implementation applying the equivalence constraint theory to languages that have significantly different grammar structures, such as English and Mandarin, for sentence generation. We also show the effectiveness of our neural-based approach in generating new code-switching sentences compared to the equivalence constraint and SeqGAN (Garg et al., 2018).

- We thoroughly analyze our generation results and further examine how our model identifies code-switching points to show its interpretability.

2 Generating Code-Switching Data

In this section, we describe our proposed model to generate code-switching sentences using a pointer-generator network. Then, we briefly list the assumptions of the equivalence constraint (EC) theory, and explain our application of EC theory for sentence generation. We call the dominant language the matrix language (L_1) and the inserted language the embedded language (L_2), following the definitions from Myers-Scotton (2001). Let us define $Q = \{Q_1, ..., Q_T\}$ as a set of L_1 sentences and $E = \{E_1, ..., E_T\}$ as a set of L_2 sentences with T number of sentences, where each $Q_t = \{q_{1,t}, ..., q_{m,t}\}$ and $E_t = \{e_{1,t}, ..., e_{n,t}\}$ are sentences with m and n words. E is the corresponding parallel sentences of Q.

2.1 Pointer-Gen

Initially, Pointer-Gen was proposed to learn when to copy words directly from the input to the output in text summarization, and they have since been successfully applied to other natural language processing tasks, such as comment generation (Lin et al., 2019). The Pointer-Gen leverages the information from the input to ensure high-quality generation, especially when the output sequence consists of elements from the input sequence, such as code-switching sequences.

We propose to use Pointer-Gen by leveraging parallel monolingual sentences to generate code-switching sentences. The approach is depicted in Figure 1. The pointer-generator model is trained from concatenated sequences of parallel sentences (Q,E) to generate code-switching sentences, constrained by code-switching texts. The words of the input are fed into the encoder. We use a bidirectional long short-term memory (LSTM), which, produces hidden state h_t in each step t. The decoder is a unidirectional LSTM receiving the word embedding of the previous word. For each decoding step, a generation probability $p_{gen} \in [0,1]$ is calculated, which weights the probability of generating words from the vocabulary, and copying words from the source text. p_{gen} is a soft gating probability to decide whether to generate the next token from the decoder or to copy the word from the input instead. The attention distribution a_t is a standard attention with general scoring (Luong et al., 2015). It considers all encoder hidden states to derive the context vector. The vocabulary distribution $P_{voc}(w)$ is calculated by concatenating the decoder state s_t and the context vector h_t^*:

$$p_{gen} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{ptr}), \quad (1)$$

where w_{h^*}, w_s, and w_x are trainable parameters and b_{ptr} is the scalar bias. The vocabulary distribution $P_{voc}(w)$ and the attention distribution a^t are weighted and summed to obtain the final distribution $P(w)$, which is calculated as follows:

$$P(w) = p_{gen} P_{voc}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t. \quad (2)$$

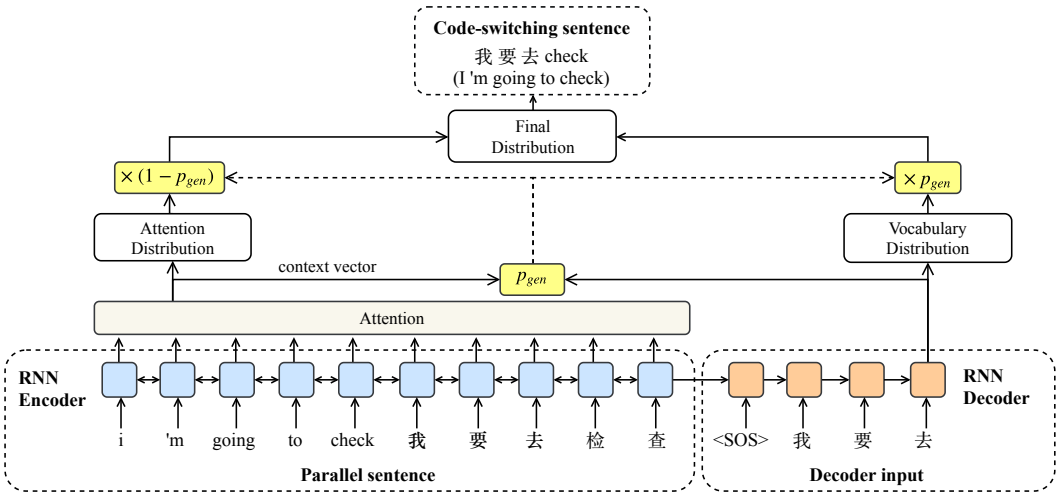


Figure 1: **Pointer-Gen** model, which includes an RNN encoder and RNN decoder. The parallel sentence is the input of the encoder, and in each decoding step, the decoder generates a new token.

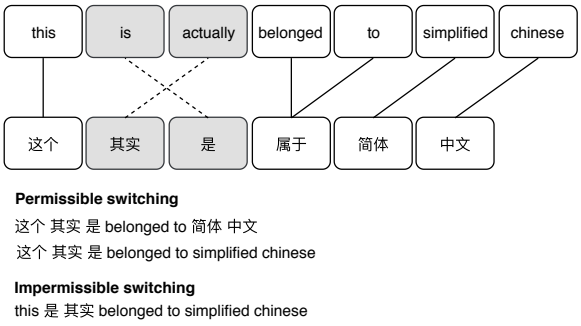


Figure 2: Example of equivalence constraint (Li and Fung, 2012). Solid lines show the alignment between the matrix language (top) and the embedded language (bottom). The dotted lines denote impermissible switching.

We use a beam search to select the N-best code-switching sentences.

2.2 Equivalence Constraint

Studies on the EC (Poplack, 1980, 2013) show that code-switching only occurs where it does not violate the syntactic rules of either language. An example of a English-Mandarin mixed-language sentence generation is shown in Figure 2, where EC theory does not allow the word "其实" to come after "是" in Chinese, or the word "is" to come after "actually". Pratapa et al. (2018) apply the EC in English-Spanish language modeling with a strong assumption. We are working with English and Mandarin, which have distinctive grammar structures (e.g., part-of-speech tags), so applying a constituency parser would give us erroneous results. Thus, we simplify sentences into a linear structure, and we allow lexical substitution on non-crossing alignments between parallel sentences. Alignments between an L_1 sentence Q_t and an L_2 sentence E_t comprise a source vector with indices $u_t = \{a_1, a_2, ..., a_m\} \in \mathbb{W}^m$ that has a corresponding target vector $v_t = \{b_1, b_2, ..., b_m\} \in \mathbb{W}^m$, where u is a sorted vector of indices in an ascending order. The alignment between a_i and b_i does not satisfy the constraint if there exists a pair of a_j and b_j, where ($a_i < a_j$, and $b_i > b_j$) or ($a_i > a_j$, and $b_i < b_j$). If the switch occurs at this point, it changes the grammatical order in both languages; thus, this switch is not acceptable. During the generation step, we allow any switches that do not violate the constraint. We propose to generate synthetic code-switching data by the following steps:

1. Align the L_1 sentences Q and L_2 sentences E using `fast_align`[2] (Dyer et al., 2013). We use the mapping from the L_1 sentences to the L_2 sentences.

2. Permute alignments from step (1) and use them to generate new sequences by replacing the phrase in the L_1 sentence with the aligned phrase in the L_2 sentence.

3. Evaluate generated sequences from step (2) if they satisfy the EC theory.

3 End-to-End Code-Switching ASR

To show the effectiveness of our proposed method, we build a transformer-based end-to-end code-

[2]The code implementation can be found at https://github.com/clab/fast_align.

switching ASR system. The end-to-end ASR model accepts a spectrogram as the input, instead of log-Mel filterbank features (Zhou et al., 2018), and predicts characters. It consists of N layers of an encoder and decoder. Convolutional layers are added to learn a universal audio representation and generate input embedding. We employ multi-head attention to allow the model to jointly attend to information from different representation subspaces at a different position.

For proficiency in recognizing individual languages, we train a multilingual ASR system trained from monolingual speech. The idea is to use it as a pretrained model and transfer the information while training the model with code-switching speech. This is an effective method to initialize the parameters of low-resource ASR such as code-switching. The catastrophic forgetting issue arises when we train one language after the other. Therefore, we solve the issue by applying a multi-task learning strategy. We jointly train speech from both languages by taking the same number of samples for each language in every batch to keep the information of both tasks.

In the inference time, we use beam search, selecting the best sub-sequence scored using the softmax probability of the characters. We define $P(Y)$ as the probability of the sentence. We incorporate language model probability $p_{lm}(Y)$ to select more natural code-switching sequences from generation candidates. A word count is added to avoid generating very short sentences. $P(Y)$ is calculated as follows:

$$P(Y) = \alpha P_{trans}(Y|X) + \beta p_{lm}(Y) + \gamma \sqrt{wc(Y)} \quad (3)$$

where α is the parameter to control the decoding probability from the probability of characters from the decoder $P_{trans}(Y|X)$, β is the parameter to control the language model probability $p_{lm}(Y)$, and γ is the parameter to control the effect of the word count $wc(Y)$.

4 Experiments

4.1 Data Preparation

We use speech data from SEAME Phase II, a conversational English-Mandarin Chinese code-switching speech corpus that consists of spontaneously spoken interviews and conversations (Nanyang Technological University, 2015). We split the corpus following information from Winata et al. (2018a). The details are depicted in Table 1. We tokenize words using the Stanford NLP toolkit (Manning et al., 2014). For monolingual speech datasets, we use HKUST (Liu et al., 2006), comprising spontaneous Mandarin Chinese telephone speech recordings, and Common Voice, an open-accented English dataset collected by Mozilla.[3] We split Chinese words into characters to avoid word boundary issues, similarly to Garg et al. (2018). We generate L_1 sentences and L_2 sentences by translating the training set of SEAME Phase II into English and Chinese using the Google NMT system (To enable reproduction of the results, we release the translated data).[4] Then, we use them to generate 270,531 new pieces of code-switching data, which is thrice the number of the training set. Table 2 shows the statistics of the new generated sentences. To calculate the complexity of our real and generated code-switching corpora, we use the following measures:

Switch-Point Fraction (SPF) This measure calculates the number of switch-points in a sentence divided by the total number of word boundaries (Pratapa et al., 2018). We define "switch-point" as a point within the sentence at which the languages of words on either side are different.

Code Mixing Index (CMI) This measure counts the number of switches in a corpus (Gambäck and Das, 2014). At the utterance level, it can be computed by finding the most frequent language in the utterance and then counting the frequency of the words belonging to all other languages present. We compute this metric at the corpus level by averaging the values for all the sentences in a corpus. The computation is shown as follows:

$$C_u(x) = \frac{N(x) - max(\ell_i \in \ell\{t_{\ell_i}(x)\}) + P(x)}{N(x)}, \quad (4)$$

where $N(x)$ is the number of tokens of utterance x, t_{ℓ_i} is the tokens in language ℓ_i, and $P(x)$ is the number of code-switching points in utterance x. We compute this metric at the corpus-level by averaging the values for all sentences.

[3]The dataset is available at https://voice.mozilla.org/.

[4]We have attached the translated data in the Supplementary Materials.

4.2 LM Training Strategy Comparison

We generate code-switching sentences using three methods: EC theory, SeqGAN (Garg et al., 2018), and Pointer-Gen. To find the best way of leveraging the generated data, we compare different training strategies as follows:

(1) rCS, **(2a)** EC, **(2b)** SeqGAN,
(2c) Pointer-Gen, **(3a)** EC & rCS,
(3b) SeqGAN & rCS, **(3c)** Pointer-Gen & rCS
(4a) EC → rCS **(4b)** SeqGAN → rCS,
(4c) Pointer-Gen → rCS

(1) is the baseline, training with real code-switching data. **(2a–2c)** train with only augmented data. **(3a–3c)** train with the concatenation of augmented data with rCS. **(4a–4c)** run a two-step training, first training the model only with augmented data and then fine-tuning with rCS. Our early hypothesis is that the results from **(2a)** and **(2b)** will not be as good as the baseline, but when we combine them, they will outperform the baseline. We expect the result of **(2c)** to be on par with **(1)**, since Pointer-Gen learns patterns from the rCS dataset, and generates sequences with similar code-switching points.

4.3 Experimental Setup

In this section, we present the settings we use to generate code-switching data, and train our language model and end-to-end ASR.

Pointer-Gen The pointer-generator model has 500-dimensional hidden states. We use 50k words as our vocabulary for the source and target. We optimize the training by Stochastic Gradient Descent with an initial learning rate of 1.0 and decay of 0.5. We generate the three best sequences using beam search with five beams, and sample 270,531 sentences, thrice the amount of the code-switched training data.

EC We generate 270,531 sentences, thrice the amount of the code-switched training data. To make a fair comparison, we limit the number of switches to two for each sentence to get a similar number of code-switches (SPF and CMI) to Pointer-Gen.

SeqGAN We implement the SeqGAN model using a PyTorch implementation[5], and use our best trained LM baseline as the generator in SeqGAN. We sample 270,531 sentences from the generator, thrice the amount of the code-switched training data (with a maximum sentence length of 20).

[5]To implement SeqGAN, we use code from https://github.com/suragnair/seqGAN.

	Train	Dev	Test
# Speakers	138	8	8
# Duration (hr)	100.58	5.56	5.25
# Utterances	90,177	5,722	4,654
# Tokens	1.2M	65K	60K
CMI	0.18	0.22	0.19
SPF	0.15	0.19	0.17

Table 1: Data statistics of SEAME Phase II. The dataset is split by speaker ID.

	EC	SeqGAN	Pointer-Gen
# Utterances	270,531	270,531	270,531
# Words	3,040,202	2,981,078	2,922,941
new unigram	13.63%	34.67%	4.67%
new bigram	69.43%	80.33%	46.57%
new trigram	99.73%	141.56%	69.38%
new four-gram	121.04%	182.89%	85.07%
CMI	0.25	0.13	0.25
SPF	0.17	0.2	0.17

Table 2: Statistics of the generated data. The table shows the number of utterances and words, code-switches ratio, and percentage of new n-grams.

LM In this work, we focus on sentence generation, so we evaluate our data with the same two-layer LSTM LM for comparison. It is trained using a two-layer LSTM with a hidden size of 200 and unrolled for 35 steps. The embedding size is equal to the LSTM hidden size for weight tying (Press and Wolf, 2017). We optimize our model using SGD with an initial learning rate of 20. If there is no improvement during the evaluation, we reduce the learning rate by a factor of 0.75. In each step, we apply a dropout to both the embedding layer and recurrent network. The gradient is clipped to a maximum of 0.25. We optimize the validation loss and apply an early stopping procedure after five iterations without any improvements. In the fine-tuning step of training strategies **(4a–4c)**, the initial learning rate is set to 1.

End-to-end ASR We convert the inputs into normalized frame-wise spectrograms from 16-kHz audio. Our transformer model consists of two encoder and decoder layers. An Adam optimizer and Noam warmup are used for training with an initial learning rate of 1e-4. The model has a hidden size of 1024, a key dimension of 64,

Training Strategy	Overall		en-zh		zh-en		en-en		zh-zh	
	valid	test	valid	test	valid	test	valid	test	valid	test
Only real code-switching data										
(1) rCS	72.89	65.71	7411.42	7857.75	120.41	130.21	29.31	29.61	244.88	246.71
Only generated data										
(2a) EC	115.98	96.54	32865.62	30580.89	107.22	109.10	28.24	28.2	1893.77	1971.68
(2b) SeqGAN	252.86	215.17	33719	37119.9	174.2	187.5	91.07	88	1799.74	1783.71
(2c) Pointer-Gen	72.78	64.67	7055.59	7473.68	119.56	133.39	27.77	27.67	234.16	235.34
Concatenate generated data with real code-switching data										
(3a) EC & rCS	70.33	62.43	8955.79	9093.01	130.92	139.06	26.49	26.28	227.57	242.30
(3b) SeqGAN & rCS	77.37	69.58	8477.44	9350.73	134.27	143.41	30.64	30.81	260.89	264.28
(3c) Pointer-Gen & rCS	68.49	61.57	7146.08	7667.82	127.50	139.06	26.75	26.96	218.27	226.60
Pretrain with generated data and fine-tune with real code-switching data										
(4a) EC → rCS	68.46	61.42	8200.78	8517.29	101.15	107.77	25.49	25.78	247.3	258.95
(4b) SeqGAN → rCS	70.61	64.03	6950.02	7694.2	114.82	122.84	28.5	28.73	236.94	244.62
(4c) Pointer-Gen → rCS	**66.08**	**59.74**	**6620.76**	**7172.42**	**114.53**	**127.12**	**26.36**	**26.40**	**216.02**	**222.49**

Table 3: Results of perplexity (PPL) on a valid set and test set for different training strategies. We report the overall PPL, and code-switching points **(en-zh)** and **(zh-en)**, as well as the monolingual segments PPL **(en-en)** and **(zh-zh)**.

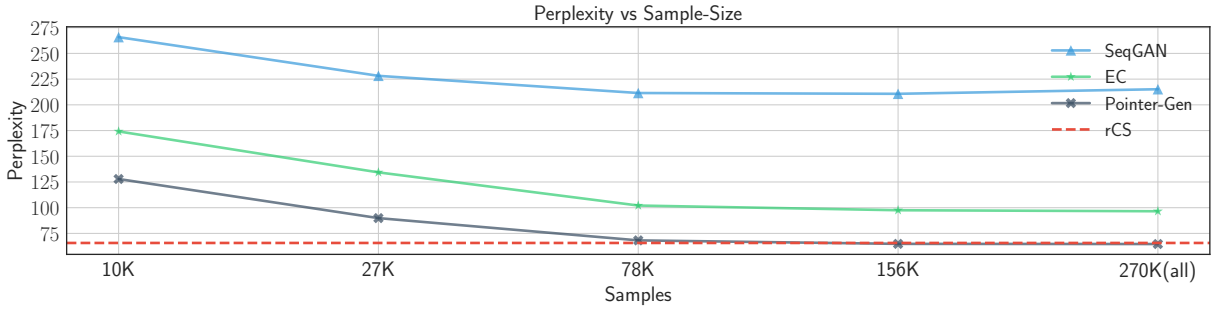


Figure 3: Results of perplexity (PPL) on different numbers of generated samples. The graph shows that Pointer-Gen attains a close performance to the real training data, and outperforms *SeqGAN* and *EC*.

and a value dimension of 64. The training data are randomly shuffled every epoch. Our character set is the concatenation of English letters, Chinese characters found in the corpus, spaces, and apostrophes. In the multilingual ASR pretraining, we train the model for 18 epochs. Since the sizes of the datasets are different, we over-sample the smaller dataset. The fine-tuning step takes place after the pretraining using code-switching data. In the inference time, we explore the hypothesis using beam search with eight beams and a batch size of 1.

4.4 Evaluation Metrics

We employ the following metrics to measure the performance of our models.

Token-level Perplexity (PPL) For the LM, we calculate the PPL of characters in Mandarin Chinese and words in English. The reason is that some Chinese words inside the SEAME corpus are not well tokenized, and tokenization results are not consistent. Using characters instead of words in Chinese can alleviate word boundary issues. The PPL is calculated by taking the exponential of the sum of losses. To show the effectiveness of our approach in calculating the probability of the switching, we split the perplexity computation into monolingual segments **(en-en)** and **(zh-zh)**, and code-switching segments **(en-zh)** and **(zh-en)**.

Character Error Rate (CER) For our ASR, we compute the overall CER and also show the individual CERs for Mandarin Chinese **(zh)** and English **(en)**. The metric calculates the distance of two sequences as the *Levenshtein Distance*.

5 Results & Discussion

LM In Table 3, we can see the perplexities of the test set evaluated on different training strategies. Pointer-Gen consistently performs better than state-of-the-art models such as *EC* and *SeqGAN*. Comparing the results of models trained using only generated samples, **(2a-2b)** leads to

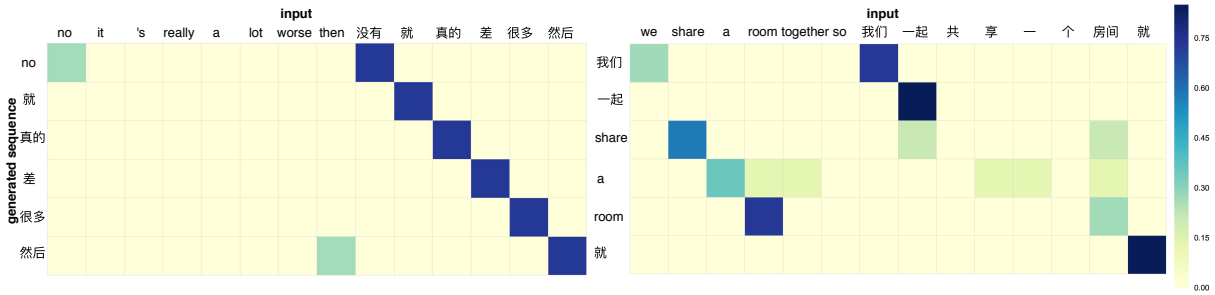


Figure 4: The visualization of pointer-generator attention weights on input words in each time-step during the inference time. The y-axis indicates the generated sequence, and the x-axis indicates the word input. In this figure, we show the code-switching points when our model attends to words in the L_1 and L_2 sentences: **left:** ("no","没有") and ("then","然后"), **right:** ("we","我们"), ("share", "一起") and ("room","房间").

the undesirable results that are also mentioned by Pratapa et al. (2018), but it does not apply to Pointer-Gen (2c). We can achieve a similar results with the model trained using only real code-switching data, rCS. This demonstrates the quality of our data generated using Pointer-Gen. In general, combining any generated samples with real code-switching data improves the language model performance for both code-switching segments and monolingual segments. Applying concatenation is less effective than the two-step training strategy. Moreover, applying the two-step training strategy achieves the state-of-the-art performance.

As shown in Table 2, we generate new n-grams including code-switching phrases. This leads us to a more robust model, trained with both generated data and real code-switching data. We can see clearly that Pointer-Gen-generated samples have a distribution more similar to the real code-switching data compared with *SeqGAN*, which shows the advantage of our proposed method.

Effect of Data Size To understand the importance of data size, we train our model with different amounts of generated data. Figure 3 shows the PPL of the models with different amounts of generated data. An interesting finding is that our model trained with only 78K samples of Pointer-Gen data (same number of samples as rCS) achieves a similar PPL to the model trained with only rCS, while SeqGAN and EC have a significantly higher PPL. We can also see that 10K samples of Pointer-Gen data is as good as 270K samples of EC data. In general, the number of samples is positively correlated with the improvement in performance.

Model	Overall	en	zh
Baseline	34.40%	41.79%	35.94%
+ Pre-training	32.76%	40.06%	32.44%
+ LM (rCS)	32.25%	39.45%	31.90%
+ LM (Pointer-Gen → rCS)	**31.07%**	**38.39%**	**30.85%**

Table 4: ASR evaluation, showing the performance on all sequences (Overall), English segments (en), and Mandarin Chinese segments (zh).

ASR Evaluation We evaluate our proposed sentence generation method on an end-to-end ASR system. Table 4 shows the CER of our ASR systems, as well as the individual CER on each language. Based on the experimental results, pre-training is able to reduce the error rate by 1.64%, as it corrects the spelling mistakes in the prediction. After we add LM (rCS) to the decoding step, the error rate can be reduced to 32.25%. Finally, we replace the LM with LM (Pointer-Gen → rCS), and it further decreases the error rate by 1.18%.

Model Interpretability We can interpret a Pointer-Gen model by extracting its attention matrices and then analyzing the activation scores. We show the visualization of the attention weights in Figure 4. The square in the heatmap corresponds to the attention score of an input word. In each time-step, the attention scores are used to select words to be generated. As we can observe in the figure, in some cases, our model attends to words that are translations of each other, for example, the words ("no","没有"), ("then","然后") , ("we","我们"), ("share", "一起"), and ("room","房间"). This indicates the model can identify code-switching points, word alignments, and translations without being given any explicit information.

rCS		Pointer-Gen		
POS tags	**ratio**	**POS tags**	**ratio**	**examples**
English				
NN (noun)	56.16%	NN (noun)	55.45%	那 个 **consumer** 是 不 (that consumer is not)
RB (adverb)	10.34%	RB (adverb)	10.14%	okay **so** 其 实 (okay so its real)
JJ (adjective)	7.04%	JJ (adjective)	7.16%	我 很 **jealous** 的 每 次 (i am very jealous every time)
VB (verb)	5.88%	VB (verb)	5.89%	**compared** 这 个 (compared to this)
Chinese				
VV (other verbs)	23.77%	VV (other verbs)	23.72%	讲 的 要 用 microsoft word (i want to use microsoft word)
M (measure word)	16.83%	M (measure word)	16.49%	我 们 有 这 个 god of war (we have this god of war)
DEG (associative)	9.12%	DEG (associative)	9.13%	我 们 的 result (our result)
NN (common noun)	9.08%	NN (common noun)	8.93%	我 应 该 不 会 讲 话 because intimidated by another (i shouldn't talk because intimidated by another)

Table 5: The most common English and Mandarin Chinese part-of-speech tags that trigger code-switching. We report the frequency ratio from **Pointer-Gen**-generated sentences compared to the real code-switching data. We also provide an example for each POS tag.

Code-Switching Patterns Table 5 shows the most common English and Mandarin Chinese POS tags that trigger code-switching. The distribution of word triggers in the Pointer-Gen data are similar to the real code-switching data, indicating our model's ability to learn similar code-switching points. Nouns are the most frequent English word triggers. They are used to construct an optimal interaction by using cognate words and to avoid confusion. Also, English adverbs such as "then" and "so" are phrase or sentence connectors between two language phrases for intra-sentential and inter-sentential code-switching. On the other hand, Chinese transitional words such as the measure word "个" or associative word "的" are frequently used as inter-lingual word associations.

6 Related Work

Code-switching language modeling research has been focused on building a model that handles mixed-language sentences and on generating synthetic data to solve the data scarcity issue. The first statistical approach using a linguistic theory was introduced by Li and Fung (2012), who adapted the EC on monolingual sentence pairs during the decoding step of an ASR system. Ying and Fung (2014) implemented a functional-head constraint lattice parser with a weighted finite-state transducer to reduce the search space on a code-switching ASR system. Then, Adel et al. (2013a) extended recurrent neural networks (RNNs) by adding POS information to the input layer and a factorized output layer with a language identifier. The factorized RNNs were also combined with an n-gram backoff model using linear interpolation (Adel et al., 2013b), and syntactic and semantic features were added to them (Adel et al., 2015). Baheti et al. (2017) adapted an effective curriculum learning by training a network with monolingual corpora of two languages, and subsequently trained on code-switched data. A further investigation of EC and curriculum learning showed an improvement in English-Spanish language modeling (Pratapa et al., 2018), and a multi-task learning approach was introduced to train the syntax representation of languages by constraining the language generator (Winata et al., 2018a). Garg et al. (2018) proposed to use SeqGAN (Yu et al., 2017) for generating new mixed-language sequences. Winata et al. (2018b) leveraged character representations to address out-of-vocabulary words in the code-switching named entity recognition. Finally, Winata et al. (2019) proposed a method to represent code-switching sentence using language-agnostic meta-representations.

7 Conclusion

We propose a novel method for generating synthetic code-switching sentences using Pointer-Gen by learning how to copy words from parallel cor-

pora. Our model can learn code-switching points by attending to input words and aligning the parallel words, without requiring any word alignments or constituency parsers. More importantly, it can be effectively used for languages that are syntactically different, such as English and Mandarin Chinese. Our language model trained using outperforms equivalence constraint theory-based models. We also show that the learned language model can be used to improve the performance of an end-to-end automatic speech recognition system.

Acknowledgments

This work has been partially funded by ITF/319/16FP and MRP/055/18 of the Innovation Technology Commission, the Hong Kong SAR Government, and School of Engineering Ph.D. Fellowship Award, the Hong Kong University of Science and Technology, and RDC 1718050-0 of EMOS.AI. We sincerely thank the three anonymous reviewers for their insightful comments on our paper.

References

Heike Adel, Ngoc Thang Vu, Katrin Kirchhoff, Dominic Telaar, and Tanja Schultz. 2015. Syntactic and semantic features for code-switching factored language models. *IEEE Transactions on Audio, Speech, and Language Processing*, 23(3):431–440.

Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz. 2013a. Recurrent neural network language modeling for code switching conversational speech. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8411–8415. IEEE.

Heike Adel, Ngoc Thang Vu, and Tanja Schultz. 2013b. Combination of recurrent neural networks and factored language models for code-switching language modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 206–211.

Ashutosh Baheti, Sunayana Sitaram, Monojit Choudhury, and Kalika Bali. 2017. Curriculum design for code-switching: Experiments with language identification and language modeling with deep neural networks. *Proceedings of ICON*, pages 65–74.

Hedi M Belazi, Edward J Rubin, and Almeida Jacqueline Toribio. 1994. Code switching and x-bar theory: The functional head constraint. *Linguistic inquiry*, pages 221–237.

Susan Berk-Seligson. 1986. Linguistic constraints on intrasentential code-switching: A study of spanish/hebrew bilingualism. *Language in society*, 15(3):313–348.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648. Association for Computational Linguistics.

Björn Gambäck and Amitava Das. 2014. On measuring the complexity of code-mixing. In *Proceedings of the 11th International Conference on Natural Language Processing, Goa, India*, pages 1–7. Citeseer.

Saurabh Garg, Tanmay Parekh, and Preethi Jyothi. 2018. Code-switched language models using dual rnns and same-source pretraining. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3078–3083.

Ying Li and Pascale Fung. 2012. Code-switch language model with inversion constraints for mixed language speech recognition. *Proceedings of COLING 2012*, pages 1671–1680.

Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2019. Learning comment generation by leveraging user-generated data. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7225–7229. IEEE.

Yi Liu, Pascale Fung, Yongsheng Yang, Christopher Cieri, Shudong Huang, and David Graff. 2006. Hkust/mts: A very large scale mandarin telephone speech corpus. In *Chinese Spoken Language Processing*, pages 724–735. Springer.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Carol Myers-Scotton. 2001. The matrix language frame model: Development and responses. *Trends in Linguistics Studies and Monographs*, 126:23–58.

Universiti Sains Malaysia Nanyang Technological University. 2015. Mandarin-english code-switching in south-east asia ldc2015s04. web download. philadelphia: Linguistic data consortium.

Carol W Pfaff. 1979. Constraints on language mixing: intrasentential code-switching and borrowing in spanish/english. *Language*, pages 291–318.

Shana Poplack. 1978. *Syntactic structure and social function of code-switching*, volume 2. Centro de Estudios Puertorriqueños,[City University of New York].

Shana Poplack. 1980. Sometimes i'll start a sentence in spanish y termino en espanol: toward a typology of code-switching1. *Linguistics*, 18(7-8):581–618.

Shana Poplack. 2013. "sometimes i'll start a sentence in spanish y termino en español": Toward a typology of code-switching. *Linguistics*, 51(Jubilee):11–14.

Adithya Pratapa, Gayatri Bhat, Monojit Choudhury, Sunayana Sitaram, Sandipan Dandapat, and Kalika Bali. 2018. Language modeling for code-mixing: The role of linguistic theory based synthetic data. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1543–1553.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 157–163.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Genta Indra Winata, Zhaojiang Lin, and Pascale Fung. 2019. Learning multilingual meta-embeddings for code-switching named entity recognition. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 181–186.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018a. Code-switching language modeling using syntax-aware multi-task learning. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 62–67. Association for Computational Linguistics.

Genta Indra Winata, Chien-Sheng Wu, Andrea Madotto, and Pascale Fung. 2018b. Bilingual character representation for efficiently addressing out-of-vocabulary words in code-switching named entity recognition. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 110–114.

LI Ying and Pascale Fung. 2014. Language modeling with functional head constraint for code switching speech recognition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 907–916.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Shiyu Zhou, Linhao Dong, Shuang Xu, and Bo Xu. 2018. Syllable-based sequence-to-sequence speech recognition with the transformer in mandarin chinese. In *Interspeech*.

Unsupervised Neural Machine Translation with Future Rewarding

Xiangpeng Wei[1,2], **Yue Hu**[1,2]*, **Luxi Xing**[1,2], **Li Gao**[3]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Platform & Content Group, Tencent, Beijing, China
{weixiangpeng, huyue, xingluxi}@iie.ac.cn
leolgao@tencent.com

Abstract

In this paper, we alleviate the local optimality of back-translation by learning a policy (takes the form of an encoder-decoder and is defined by its parameters) with future rewarding under the reinforcement learning framework, which aims to optimize the global word predictions for unsupervised neural machine translation. To this end, we design a novel reward function to characterize high-quality translations from two aspects: n-gram matching and semantic adequacy. The n-gram matching is defined as an alternative for the discrete BLEU metric, and the semantic adequacy is used to measure the adequacy of conveying the meaning of the source sentence to the target. During training, our model strives for earning higher rewards by learning to produce grammatically more accurate and semantically more adequate translations. Besides, a variational inference network (VIN) is proposed to constrain the corresponding sentences in two languages have the same or similar latent semantic code. On the widely used WMT'14 English-French, WMT'16 English-German and NIST Chinese-to-English benchmarks, our models respectively obtain 27.59/27.15, 19.65/23.42 and 22.40 BLEU points without using any labeled data, demonstrating consistent improvements over previous unsupervised NMT models.

1 Introduction

Neural Machine Translation (Sutskever et al., 2014; Bahdanau et al., 2015) directly models the entire translation process through training an encoder-decoder model that has achieved remarkable performance (Wu et al., 2016; Gehring et al., 2017; Vaswani et al., 2017) when provided with massive amounts of parallel corpora. However, the lack of large-scale parallel data is a serious problem for the vast majority of language pairs.

As a result, several works have recently tried to get rid of the dependence on parallel corpora using unsupervised setting, in which the NMT model only has access to two independent monolingual corpora with one for each language (Lample et al., 2018a; Artetxe et al., 2018b; Yang et al., 2018). Among these works, the encoder and decoder act as a standard auto-encoder (AE) that are trained to reconstruct the inputs from their noised versions. Due to the lack of cross-language signals, unsupervised NMT usually requires pseudo parallel data generated with the back-translation method for achieving the final goal of translating between source and target languages.

Back-translation typically uses beam search (Sennrich et al., 2016a) or just greedy search (Lample et al., 2018a,b) to generate synthetic sentences. Both are approximate algorithms to identify the maximum a posteriori (MAP) output, i.e. the sentence with the highest estimated probability given an input. Although back-translation with MAP prediction has been proved to be successful, it suffers from several apparent issues when trained with maximum likelihood estimation (MLE) only, including exposure bias and loss-evaluation mismatch. Thus, this method often fails to produce the optimal synthetic sentences for the subsequent training.

In this paper, we address the problem mentioned above with future rewarding for unsupervised NMT. The basic idea is to model the future direction of a translation and optimize the global word predictions under the policy gradient reinforcement learning framework. More concretely, we sample N translations via the policy for each input sentence and build a new objective function by combining the cross-entropy loss used in prior works with sequence-level rewards from policy gradient reinforcement learning. We consider the sequence-level reward from two aspects: 1) n-

*Corresponding Author.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 281–290
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

gram matching, which is the precision or recall of all sub-sequences of 1, 2, 3 and 4 tokens in generated sequence and is responsible for measuring the accuracy of surface word predictions; 2) semantic adequacy, which is the similarity between the underlying semantic representations of the generated translation and the input sentence. These two aspects of rewards are inspired by the general criteria of what properties a high-quality translation should have and are complementary to each other. Additionally, a variational inference network (VIN) is proposed to model the underlying semantics of monolingual sentences explicitly. It is used to map the source and target languages into a shared semantic space during auto-encoding, as well as constrain the sentences and their translated counterparts have the same or similar semantic code during cross-language training.

The major contributions of this paper can be summarized as follows:

- We propose a novel learning paradigm for unsupervised NMT that models future rewards to optimize the global word predictions via policy gradient reinforcement learning. To enforce the underlying semantic space, we introduce a VIN into our model.
- We introduce an effective reward function that jointly accounts for the n-gram matching and the semantic adequacy of generated translations.
- We conduct extensive experiments on English-French, English-German and NIST Chinese-to-English translation tasks. Experimental results show that the proposed approach achieves significant improvements across different language pairs.

2 Unsupervised Neural Machine Translation

In this section, we first describe the composition of the introduced model and then give details of the newly proposed unsupervised training method.

2.1 Model Composition

The introduced translation model consists of six components: including two encoders with sharing last few layers, two completely independent decoders with one for each language, and two newly introduced VINs with one for each language. For the encoders and decoders, we follow

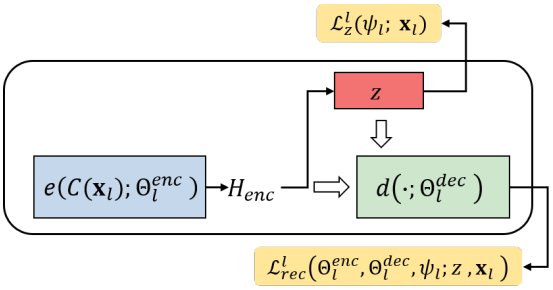


Figure 1: Illustration of Variational Denosing Auto-Encoding. The newly introduced VIN is highlighted in red. Two aspects of losses are respectively abbreviated as $\mathcal{L}_z^l$ and $\mathcal{L}_{rec}^l$.

the recently emerged Transformer (Vaswani et al., 2017). Specifically, each encoder is composed of a stack of four identical layers, and each layer consists of a multi-head self-attention sub-layer and a fully connected feed-forward sub-layer. The encoders of the source and target languages are respectively parameterized as Θ_{src}^{enc} and Θ_{tgt}^{enc}, and the encoding operation is denoted as $e(\mathbf{x}_l; \Theta_l^{enc})$, $\mathbf{x}_l$ is the input sequence of word embeddings, $l \in \{src, tgt\}$. The decoders are also composed of four identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack, the details we refer the reader to (Vaswani et al., 2017). Similar to encoders, we denote source decoder as Θ_{src}^{dec}, target decoder as Θ_{tgt}^{dec}, and decoding operation as $d(\mathbf{x}_l; \Theta_l^{dec})$, $l \in \{src, tgt\}$. For VINs, each of them is composed of a standard Gaussian distribution $\mathcal{N}(0, \mathbf{1})$ as the *prior*, and a *neural posterior* that is implemented as feed-forward neural network and parameterized by ψ_l, $l \in \{src, tgt\}$.

In this work, the entire model is trained in an unsupervised manner by optimizing two objectives: 1) variational denoising auto-encoding; 2) cross-language training with future rewarding.

2.2 Variational Denoising Auto-Encoding

Firstly, two auto-encoders are respectively trained to learn to reconstruct their inputs. In this form, each encoder should learn to compose the input sentence of its corresponding language, and each decoder is expected to learn to recover the original input sentence from this composition. However, without any constraint, the auto-encoder would make very literal word-by-word copies, without capturing any internal structure of the input sentence involved. To address this issue, prior works

often adapt the same strategy as Denosing Auto-Encoding (DAE) (Vincent et al., 2008), and add some noise to the input sentences (Hill et al., 2016). As shown in Figure 1, we augment the DAE with a variational inference network (VIN) to model underlying semantics of monolingual sentences explicitly, which assumes that there exists a latent variable $\mathbf{z}$ from this semantic space. And this variable, together with the noised input sentence, guides the decoding process. With this assumption, we define the objective function of reconstruction as follow:

$$\mathcal{L}^l_{rec} = \log P_{\Theta_{l\to l}}(\mathbf{x}_l|\mathbf{z}, C(\mathbf{x}_l)) \quad (1)$$

where $\Theta_{l\to l} = \Theta^{enc}_l \circ \Theta^{dec}_l \circ \boldsymbol{\psi}_l$ represents the combination of Θ^{enc}_l, Θ^{dec}_l and $\boldsymbol{\psi}_l$, $l \in \{src, tgt\}$. C denotes a stochastic noise model, in which we apply the same method as in (Lample et al., 2018a).

The continuous latent variable $\mathbf{z}$, acts as the underlying semantics here, is approximated by a *neural posterior* inference network $q_{\boldsymbol{\psi}_l}(\mathbf{z}|\mathbf{x}_l)$. Following (Kingma and Welling, 2014; Kingma et al., 2014), the posterior approximation is regarded as a diagonal Gaussian $\mathcal{N}(\mu, diag(\sigma^2))$, and its mean μ and variance σ^2 are parameterized with deep neural networks. We also reparameterize $\mathbf{z}$ as a function of μ and σ (i.e., $\mathbf{z} = \mu + \sigma \odot \varepsilon$, ε is a standard Gaussian variable that plays a role of introducing noises) rather than using the standard sampling method. We aim to map source and target languages into a shared semantic space and use the following objective function for VINs:

$$\mathcal{L}^l_z = -\mathrm{KL}(q_{\boldsymbol{\psi}_l}(\mathbf{z}|\mathbf{x}_l)||\mathcal{N}(0, \mathbf{1})) \quad (2)$$

where $l \in \{src, tgt\}$. $\mathrm{KL}(Q||P)$ is the Kullback-Leibler divergence between Q and P.

We finally incorporate the auto-encoder and the VIN into an end-to-end neural network, and the overall training objective of auto-encoding is to minimize the following loss function:

$$\mathcal{L}^l_{ae} = -(\mathcal{L}^l_z + \mathcal{L}^l_{rec}) \quad (3)$$

2.3 Cross-language Training with Future Rewarding

In spite of the auto-encoding, the second objective of unsupervised NMT is to constrain the model to be able to map an input sentence from the source (target) language to the target (source) language.

Due to the lack of alignment information between two independent monolingual corpora, the back-translation (Sennrich et al., 2016a) method is used to synthetise a pseudo parallel corpus for cross-language training. More concretely, given an input sentence in one language, which can be firstly translated into the other language (i.e. use the corresponding encoder and the decoder of the other language) by applying the model in inference mode with greedy decoding. And then, the model is trained to reconstruct the original sentence from this translation. The most widely used method in previous works to train the model for sequence generation, called maximum likelihood estimation (MLE for short), it assumes that the ground-truth is provided at each step during training. The objective of MLE is defined as the maximization of the following log-likelihood:

$$\mathcal{L}^{l_1}_{mle} = \log P_{\Theta_{l_2\to l_1}}(\mathbf{x}_{l_1}|\mathbf{z}_p, \tilde{\mathbf{x}}_{l_2}) \quad (4)$$

where $\Theta_{l_2\to l_1} = \Theta^{enc}_{l_2} \circ \Theta^{dec}_{l_1} \circ \boldsymbol{\psi}_{l_2}$ represents the combination of $\Theta^{enc}_{l_2}$, $\Theta^{dec}_{l_1}$ and $\boldsymbol{\psi}_{l_2}$. $\mathbf{z}_p$ is approximated by the introduced VIN (i.e., reparameterized from the Gaussian $q_{\boldsymbol{\psi}_{l_2}}(\mathbf{z}_p|\tilde{\mathbf{x}}_{l_2})$). $\tilde{\mathbf{x}}_{l_2} = d(e(\mathbf{x}_{l_1}; \Theta^{enc}_{l_1}); \Theta^{dec}_{l_2})$ is obtained by greedy decoding in inference mode ($l_1 = src, l_2 = tgt$ or $l_1 = tgt, l_2 = src$).

2.3.1 Future Rewarding

Unfortunately, maximizing $\mathcal{L}^{l_1}_{mle}$ does not always produce the best results on discrete evaluation metrics such as BLEU (Papineni et al., 2002), as the accumulation of errors caused by exposure bias as well as the inconsistency between training and testing measurements lead to the models tend to be short-sighted. We bridge the discrepancy between training and testing modes caused by MLE through learning a policy to model future rewards, which can directly optimize the global word predictions and is made possible with reinforcement learning, as illustrated in Figure 2. To reduce the variance of the model, we use the self-critical policy gradient learning algorithm (Rennie et al., 2017).

For self-critical policy gradient learning, we produce two separate output sequences at each training iteration: $\hat{\mathbf{x}}$, the sampled translation, which is obtained by sampling from the final output probability distribution, and $\hat{\mathbf{x}}^g$, the baseline output, obtained by performing a greedy search. Thus, the objective function of cross-language training can be redefined as the expected advantages of the sampled sequence over the baseline

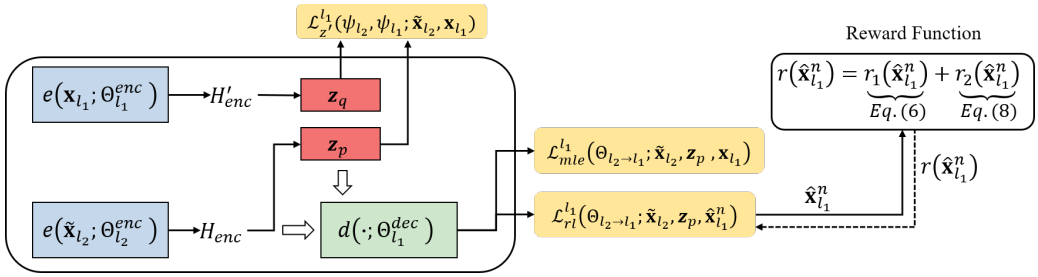


Figure 2: Illustration of the proposed method for cross-language training with future rewarding. Three aspects of losses are respectively abbreviated as $\mathcal{L}^{l_1}_{z'}$, $\mathcal{L}^{l_1}_{mle}$ and $\mathcal{L}^{l_1}_{rl}$. And $\mathcal{L}^{l_1}_{z'}$ is an auxiliary function that constrains the sentences and their translated counterparts in other language have the same or similar semantic codes.

sequence:

$$\begin{aligned}\mathcal{L}^{l_1}_{rl} =& \mathbb{E}_{P_{\Theta_{l_2\to l_1}}(\hat{\mathbf{x}}_{l_1}|\mathbf{z}_p,\tilde{\mathbf{x}}_{l_2})}[r(\hat{\mathbf{x}}_{l_1}) - r(\hat{\mathbf{x}}^g_{l_1})] \\ =& \log P_{\Theta_{l_2\to l_1}}(\hat{\mathbf{x}}_{l_1}|\mathbf{z}_p,\tilde{\mathbf{x}}_{l_2}) \times [r(\hat{\mathbf{x}}_{l_1}) - r(\hat{\mathbf{x}}^g_{l_1})]\end{aligned} \tag{5}$$

where a terminal reward r is observed after the generation reaches the end of each sentence. It is worth noting that considering a baseline reward into training objective can reduce the variance of the model. And we can see that maximizing $\mathcal{L}_{rl}$ is equivalent to maximizing the conditional likelihood of the sampled sequence $\hat{\mathbf{x}}$ if it obtains a higher reward than the baseline $\hat{\mathbf{x}}^g$, thus increasing the expected reward of our model.

2.3.2 Reward

r in Equation 5 denotes the sequence-level reward that evaluates the quality of generated translations. In this subsection, we discuss two major factors that contribute to the success of a translation, that is, n-gram matching and semantic adequacy, and describe how to approximate these factors through computable reward functions.

N-gram matching For a translation generated by a NMT model, we need to measure the accuracy of surface word predictions. For that purpose, the BLEU (Papineni et al., 2002) score is often utilized in previous works. However, the BLEU score has some undesirable properties when used for single sentences, as it was designed to be a corpus measure. Thus, we apply the smoothed version of GLEU (Wu et al., 2016) as the reward for measuring n-gram precision or recall. More concretely, given a generated translation $\hat{\mathbf{x}}_{l_1}$ in one language and the ground-truth reference $\mathbf{x}_{l_1}$, we record all sub-sequences of 1, 2, 3 and 4 tokens in $\hat{\mathbf{x}}_{l_1}$ and $\mathbf{x}_{l_1}$, and start all n-gram counts from 1 instead of 0. Then we compute a recall R_{gleu}, which is the ratio of the number of matching n-grams to the number of total n-grams in $\mathbf{x}_{l_1}$ (ground-truth), and a precision P_{gleu}, which is the ratio of the number of matching n-grams to the number of total n-grams in $\hat{\mathbf{x}}_{l_1}$ (generated output). Finally, the reward of the generated translation $\hat{\mathbf{x}}_{l_1}$ on n-gram matching is defined as:

$$r_1(\hat{\mathbf{x}}_{l_1}) = \min\{R_{gleu}, P_{gleu}\} \tag{6}$$

where r_1 ranges from zero to one and it is symmetrical when switching $\hat{\mathbf{x}}$ and $\mathbf{x}$.

Semantic adequacy We want the model can adequately convey the meaning of the source sentence to the target as much as possible. Thus, we introduce another crucial reward function that is used to measure the semantic adequacy of the generated translations. More concretely, for a generated translation $\hat{\mathbf{x}}_{l_1}$ in one language, we compute the representation of $\hat{\mathbf{x}}_{l_1}$ as:

$$\begin{aligned} e_i &= \text{TFIDF}(w_i), w_i \in \hat{\mathbf{x}}_{l_1} \\ \mathbf{w}_i &= e_i / \text{Sum}(e_1, e_2, ..., e_{T_{\hat{\mathbf{x}}_{l_1}}}) \\ c_{\hat{\mathbf{x}}_{l_1}} &= \sum_{i=1}^{T_{\hat{\mathbf{x}}_{l_1}}} \mathbf{w}_i \hat{\mathbf{x}}^i_{l_1} \end{aligned} \tag{7}$$

Identically, for the corresponding input sentence in another language, its representation $c_{\tilde{\mathbf{x}}_{l_2}}$ can be extracted from the embedding matrix $\tilde{\mathbf{x}}_{l_2}$. As the source and target word embeddings are often mapped to a shared-latent space in unsupervised NMT, we therefore can directly use the following cosine similarity as the reward for semantic adequacy:

$$r_2(\hat{\mathbf{x}}_{l_1}) = \frac{(c_{\hat{\mathbf{x}}_{l_1}}, c_{\tilde{\mathbf{x}}_{l_2}})}{\|c_{\hat{\mathbf{x}}_{l_1}}\| \cdot \|c_{\tilde{\mathbf{x}}_{l_2}}\|} \tag{8}$$

where $(,)$ indicates the dot product operation.

The final reward for a translation $\hat{\mathbf{x}}_{l_1}$ is a linear combination of the rewards discussed above:

$$r(\hat{\mathbf{x}}_{l_1}) = r_1(\hat{\mathbf{x}}_{l_1}) + r_2(\hat{\mathbf{x}}_{l_1}) \quad (9)$$

where $r_1(\hat{\mathbf{x}}_{l_1})$ and $r_2(\hat{\mathbf{x}}_{l_1})$ complement to each other and work jointly to guide the learning of our model. Note that the combination of these two aspects of rewards helps because it can prevent the cases that the generated translation with high n-gram matching but low semantic adequacy to have relatively high rewards, and vice versa.

2.3.3 Overall Objective Function

In addition to the aforementioned MLE objective function (Eq. 4) and the RL objective function (Eq. 5), there is an auxiliary function that constrains the sentences and their translated counterparts have the same or similar semantic code and is defined as:

$$\mathcal{L}_{z'}^{l_1} = -\mathrm{KL}(q_{\psi_{l_1}}(\mathbf{z}_q|\mathbf{x}_{l_1})||q_{\psi_{l_2}}(\mathbf{z}_p|\tilde{\mathbf{x}}_{l_2})) \quad (10)$$

Finally, the overall training objective of cross-language training is to minimize the following loss function with hyperparameters η:

$$\mathcal{L}_{cl}^{l_1} = -((1-\eta)(\mathcal{L}_{mle}^{l_1} + \mathcal{L}_{z'}^{l_1}) + \eta\mathcal{L}_{rl}^{l_1}) \quad (11)$$

where η is a scaling factor. In the beginning of the training $\eta = 0$, while as we move on with the training we can increase the η to slowly reduce the effect of MLE loss. And η is updated as follows:

$$\eta = \min(0.8, \max(0.0, \frac{steps - n_s}{n_e - n_s})) \quad (12)$$

where $steps$ is the global steps that the model has been updated, n_s and n_e are the start and end steps for increasing η respectively.

2.4 Training Procedure

There are two stages in the proposed unsupervised training. In the first stage, we pre-train the proposed model with denoising auto-encoding and cross-language training, until no improvement is achieved on the development set. This ensures that the model starts with a much better policy than random because now the model can focus on the good part of the search space. In the second state, we use an annealing schedule to teach the model to produce stable sequences gradually. That is, after the initial pre-training steps, we continue training the model with future rewarding. During each iteration, we perform one batch of denoising auto-encoding and cross-language training for the source as well as target languages alternately.

For model selection, we randomly extract 3000 source and target sentences to form a development set. Following (Lample et al., 2018a), we translate the source sentences to the target language and then convert the resulting sentences back to the source language. The quality of the model is then evaluated by computing the BLEU score over the original inputs and their reconstructions via this two-step translation process. The performance is finally averaged over two directions, and the selected model is the one with the highest score.

3 Experiments

We mainly evaluate the proposed approach on the widely used English-German, English-French and NIST Chinese-to-English[1] translation tasks.

3.1 Datasets

For English-French and English-German, we use 30M sentences from the WMT monolingual News Crawl datasets from years 2007 through 2017. We use the publicly available implementation of Moses[2] scripts for tokenization. Besides, we use a shared vocabulary for source and target languages with 60K subword tokens based on byte-pair encoding (Sennrich et al., 2016b). We remove sentences longer than 50 subword-tokens. Experimental results are reported on *newstest2014* for English-French translation and *newstest2016* for English-German translation. We adopt the same method as in (Lample et al., 2018b) to obtain cross-lingual embeddings.

For NIST Chinese-to-English translation, our training data consists of 1.6M sentence pairs randomly extracted from LDC corpora[3], which has been widely utilized by previous works. Similar to (Yang et al., 2018), we build the monolingual dataset by randomly shuffling the Chinese and English sentences respectively since the data set is not big enough. We set the vocabulary size to 30K for both Chinese and English. The average BLEU score over *NIST02∼06* is reported

[1]The reason that we do not conduct experiments on English-to-Chinese translation is that we do not get public test sets for English-to-Chinese.

[2]http://www.statmt.org/moses/

[3]LDC2002E18, LDC2003E07, LDC2003E14, the Hansards portion of LDC2004T07, LDC2004T08, and LDC2005T06

	en→fr	fr→en	en→de	de→en	zh→en
Existing Unsupervised NMT					
Artetxe et al. (2018b)	15.13	15.56	-	-	-
Lample et al. (2018a)	15.05	14.31	9.64	13.33	-
Yang et al. (2018)	16.97	15.58	10.86	14.62	14.52
Lample et al. (2018b), NMT	25.14	24.18	17.16	21.00	-
Wu et al. (2019)	27.56	26.90	19.55	23.29	-
Song et al. (2019)	27.41	27.09	18.21	23.37	-
This work					
MLE	25.47	24.51	17.04	21.13	18.26
(+Future Rewarding)	**27.59**	**27.15**	**19.65**	**23.42**	**22.40**

Table 1: Results of the proposed method in comparison to existing unsupervised NMT systems (BLEU).

in this paper. To pre-train cross-lingual embeddings, we utilize the monolingual corpora to train the embeddings for each language independently by using word2vec (Mikolov et al., 2013). Then we apply the public implementation[4] proposed by Artetxe et al. (2017) to map these embeddings into a shared latent space and keep the mapped embeddings fixed during training.

For NIST Chinese-to-English, we apply case-insensitive NIST BLEU computed by the script *mteval-v13a.pl* to evaluate the translation performance. For English-German and English-French, we evaluate the translation performance with the script *multi-belu.pl*.

3.2 Hyper-parameters

We set the following hyper-parameters: word embedding dimension as 512, hidden size of self-attention as 512, hidden size of fully connected layers as 1024 and the head number as 8. We share the last one layer of encoders in both languages. The dropout rate is set as 0.1, 0.3 and 0.2 during the training for En-Fr, En-De and Zh-to-En, respectively. We perform a fixed number of iterations (500K) to train each model, and set $n_s = 300$K, $n_e = 400$K, for gradually increasing the effect of future rewarding. We use the Adam optimizer with a simple learning rate schedule: we start with a learning rate of 10^{-4}, after 300K updates, we begin to halve the learning rate every 100K steps. We set the mini-batch size as 64. At decoding time, we use greedy search.

3.3 Overall Results

Our method is compared with several previous unsupervised NMT systems (Artetxe et al., 2018b; Lample et al., 2018a,b; Yang et al., 2018; Wu et al., 2019; Song et al., 2019). Although, Song et al. (2019) have achieved comparable results with supervised NMT systems with larger monolingual data (Wikipedia data) and bigger model[5], we still list the results that obtained with the same data and model as ours for fair comparison. We also consider a "Baseline" model, with the same architecture as described in Section 2.1 except for the variational inference network and is trained using MLE only. We directly copy the experimental results of previous models reported in their papers and report the BLEU scores on English-French, English-German and NIST Chinese-to-English test sets in Table 1.

As shown in Table 1, our approach achieves BLEU score of 27.59 and 27.15 on En→Fr and Fr→En translations respectively, which outperforms Lample et al. (2018b) by more than 2 BLEU points on both En→Fr and Fr→En. For the En-De, we achieve 19.65 and 23.42 BLEU scores on En→De and De→En respectively, with up to +10.09 BLEU points improvement over previous unsupervised NMT models. For the Chinese-to-English translation, the proposed method leads to a substantial improvement (up to 54%) over the previous system showed in Yang et al. (2018). Compared to baseline, our approach demonstrates significant improvements by more than 2 BLEU points over three benchmarks. These results indicate that the newly proposed training method that models future rewards to optimize global word predictions for unsupervised NMT is promising and enables the model to generate quality translations.

[4]https://github.com/artetxem/vecmap

[5]Our model can also adopt such an advanced pre-training technique, we leave this for feature work.

3.4 Analysis

In this section, we conduct some analysis over the proposed method by taking English-French translation as an example.

3.4.1 Ablation Study

To understand the importance of different components of the proposed system, we perform an ablation study by training multiple versions of our model with some missing components: the variational inference network and the future rewarding method. Results are reported in Table 2. From the table, we can see that removing the future rewarding, and the accuracy drops by 0.98/1.02 BLEU points. Without the variational inference networks, the accuracy decreases with 0.62/0.69 BLEU points. These findings demonstrate that both the future rewarding and the VIN are important, and both contribute to the improvement of translation accuracy. The more critical component is the future rewarding technology, which is vital to optimize the global word predictions.

	en-fr	fr-en
Full Model	27.59	27.15
Without VINs	26.97	26.46
Without Future Rewarding	26.61	26.13

Table 2: Ablation study of our method on English-French translation task.

3.4.2 Qualitative Comparison of Back-translating

We perform qualitative evaluation on the pseudo parallel data generated with the back-translation method. To this end, we conduct a "round-trip" translation (e.g., $src \rightarrow \tilde{tgt} \rightarrow \hat{src}$), where src and $\tilde{tgt}$ form a pseudo parallel corpus, $\hat{src}$ is the reconstruction from $\tilde{tgt}$. We explore three settings for qualitative evaluation: 1) *UNKs*, the ratio of the number of unknown words to the number of total words in $\tilde{tgt}$; 2) the average over all sentences in $\tilde{tgt}$ with respect of their semantic adequacy, denoted as *SA*; 3) the BLEU scores over the original inputs and their reconstructions, denoted as *r-BLEU*. All settings are finally averaged over two directions.

Results are shown in Table 3. The proposed training method introduces significant boosts in all of the three settings, with reducing 1.34% of unknown words, increasing the semantic adequacy by 0.088 and improving r-BLEU points by 5.85. This is in line with our expectations, as the proposed future rewarding method is not optimized to predict the next token, but rather to increase long-term reward.

	UNKs	SA	r-BLEU
Baseline	3.51%	0.794	54.23
+Future Rewarding	2.17%	0.882	60.08

Table 3: Qualitative comparison of the generated pseudo parallel sentences from the models trained with MLE only and with the proposed training method on English-French test set.

Better than Baseline	
S:	He put together a real feast for his fans to mark the occasion.
R:	Pour l'occasion, il a concocté un vrai festin pour ses fans.
B:	Il a mis en scène un vrai festin pour *son public* pour marquer le *souvenir*.
O:	Il a mis un vrai festin pour **ses fans** pour marquer la **circonstance**.
S:	Des scientifiques viennent de mettre en lumière la façon dont les mouvements de la queue d'un chien sont liés à son humeur.
R:	Scientists have shed more light on how the movements of a dog's tail are linked to its mood.
B:	Scientists *come out of light the way the movements of* the tail of a dog are linked to *his spirits*.
O:	Scientists **come to light** the way of the movements of a dog's tail are related to **its mood**.
Worse than Baseline	
S:	The recalled models were built between August 1 and September 10.
R:	Les modèles rappelés ont été construits entre le 1er août et le 10 septembre.
B:	Les modèles *rappelés* ont été construits entre le 1er août et le 10 septembre.
O:	Les modèles **de raconté** ont été construits entre le 1er août et le 10 septembre.
S:	Elles connaissent leur entreprise mieux que personne.
R:	They know their business better than anyone else.
B:	They know their *business* better than *anyone else*.
O:	They know their **company** better than **anyone**.

Table 4: Translation examples from English-French test set (English-to-French is above the dotted line and French-to-English is below the dotted line). **B**: the baseline model; **O**: our proposed model.

3.4.3 Example Translations

Table 4 shows four example translations. The first part shows examples for which the proposed model reached a higher BLEU score than the baseline model. We find that the translation produced

by the baseline model doesn't adequately convey the meaning of the source sentence to the target. By contrast, the proposed future rewarding method enables the model to generate translations that are more diversity while ensuring the meaning of the source sentences, such as "***circonstance***" and "***come to light***". The possible reason is that we apply the semantic adequacy to reward translations that have different syntax structures and expressions but share the same meaning as the ground-truth sentence. The second part contains examples where the baseline achieved better BLEU score than our model, that is, in a few cases, our model chooses inappropriate words that under the same topic as reference words.

4 Related Work

In order to reduce the exposure bias and optimize the metrics used to evaluate sequence modeling tasks (like BLEU, ROUGE or METEOR) directly, reinforcement learning (RL) has been widely used in many of recent works on machine translation (Ranzato et al., 2016; Shen et al., 2016; He et al., 2017; Bahdanau et al., 2017; Li et al., 2017), text summarization (Paulus et al., 2018; Wu and Hu, 2018; Li et al., 2018; Wang et al., 2018), dialogue generation (Li et al., 2016), and question answering (Hu et al., 2018). However, our proposed method is the first use in combination with reinforcement learning for unsupervised NMT to explicitly enhance back-translation.

Recently, motivated by the success of cross-lingual embeddings (Artetxe et al., 2016; Zhang et al., 2017; Conneau et al., 2017), several works have tried to train NMT or SMT models using unsupervised setting, in which the model only has access to unlabeled data. For example, Lample et al. (2018a) propose a model that consists of a single encoder and a single decoder for both languages, respectively responsible for encoding source and target sentences to a shared latent space and to decode from that latent space to the source or target domain. Different from (Lample et al., 2018a), Artetxe et al. (2018b) introduce a shared encoder but two independent decoders with one for each language. Both of these two works mentioned above utilize denoising auto-encoding to reconstruct their noisy inputs and incorporate back-translation into cross-language training procedure. Further, Yang et al. (2018) extend the single encoder by using two independent encoders but sharing some partial weights, which are responsible for alleviating the weakness in keeping language-specific characteristics of the shared encoder. And the entire system is fine-tuned by introducing two global GANs with one for each language. More recently, Artetxe et al. (2018a) and Lample et al. (2018b) propose an alternative approach based on phrase-based statistical machine translation, which profits from the modular architecture of SMT. In addition, Lample et al. (2018b) also introduce a novel cross-lingual embedding training method which is particularly suitable for related languages (e.g., English-French and English-German). Ren et al. (2019) introduce SMT models as posterior regularization, in which SMT and NMT models boost each other through iterative back-translation in a unified EM training algorithm. Wu et al. (2019) propose an alternative for back-translation, , extract-edit, to extract and then edit real sentences from the target monolingual corpora. Lample and Conneau (2019) and Song et al. (2019) propose to pretrain cross-lingual language models for the initialization stage of unsupervised neural machine translation, which is critical to the performance of their proposed model. In contrast to theirs, we propose an effective training method for unsupervised NMT that models future rewards to optimize the global word predictions via neural policy reinforcement learning, which can be applied to arbitrary architectures and language pairs easily.

5 Conclusion

In this paper, we have proposed a novel learning paradigm for unsupervised NMT that models future rewards to optimize the global word predictions via reinforcement learning, in which we design an effective reward function that jointly accounts for the n-gram matching and the semantic adequacy of generated translations. To constrain the corresponding sentences in two languages have the same or similar semantic code, we also introduce a variational inference network into the proposed model.

We test the proposed model on WMT'14 English-French, WMT'16 English-German and NIST Chinese-to-English translation tasks. Experiment results show that our approach leads to significant improvements over various language pairs, especially on distantly-related languages such as Chinese and English.

6 Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported by the National Key Research and Development Program of China (No. 2017YFB0803301). Yue Hu is the corresponding author.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2016. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pages 2289–2294.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 451–462.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Unsupervised statistical machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 3632–3642.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018b. Unsupervised neural machine translation. In *ICLR 2018*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR 2017*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. In *arXiv:1710.04087*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *arXiv:1705.03122*.

Di He, Hanqing Lu, Yingce Xia, Tao Qin, Liwei Wang, and Tieyan Liu. 2017. Decoding with value networks for neural machine translation. In *Advances in Neural Information Processing Systems, NIPS 2017*, pages 178–187.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of NAACL-HLT 2016*, pages 1367–1377.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 4099–4106.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems, NIPS 2014*, pages 3581–3589.

Diederik P Kingma and Max Welling. 2014. Auto–encoding variational bayes. In *ICLR 2014*.

Guillaume Lample and Alexis Conneau. 2019. Cross–lingual language model pretraining. In *arXiv:1901.07291*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018a. Unsupervised machine translation using monolingual corpora only. In *ICLR 2018*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018b. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 5039–5049.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Learning to decode for future success. In *arXiv:1701.06549*.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, ENMLP 2016*, page 1192–1202.

Piji Li, Lidong Bing, and Wai Lam. 2018. Actor-critic based training framework for abstractive summarization. In *arXiv:1803.11070*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems, NIPS 2013*, pages 3111–3119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics, ACL 2002*, pages 311–318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR 2018*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR 2016*.

Shuo Ren, Zhirui Zhang, Shujie Liu, Ming Zhou, and Shuai Ma. 2019. Unsupervised neural machine translation with SMT as posterior regularization. In *arXiv:1901.04112*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pages 1179–1195.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, pages 1715–1725.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*, pages 1683–1692.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. In *arXiv:1905.02450*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems, NIPS 2014*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *arXiv:1706.03762*.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference, ICML 2008*, page 1096–1103.

Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. In *arXiv:1805.03616*.

Jiawei Wu, Xin Wang, and William Yang Wang. 2019. Extract and edit: An alternative to back-translation for unsupervised neural machine translation. In *arXiv:1904.02331*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, and Klaus Macherey. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *arXiv:1609.08144*.

Yuxiang Wu and Baotian Hu. 2018. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 5602–5609.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Unsupervised neural machine translation with weight sharing. In *arXiv:1804.09057*.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 1959–1970.

Automatically Extracting Challenge Sets for Non-local Phenomena in Neural Machine Translation

Leshem Choshen[1] and Omri Abend[1,2]
[1]School of Computer Science and Engineering, [2] Department of Cognitive Sciences
The Hebrew University of Jerusalem
leshem.choshen@mail.huji.ac.il, oabend@cs.huji.ac.il

Abstract

We show that the state-of-the-art Transformer MT model is not biased towards monotonic reordering (unlike previous recurrent neural network models), but that nevertheless, long-distance dependencies remain a challenge for the model. Since most dependencies are short-distance, common evaluation metrics will be little influenced by how well systems perform on them. We therefore propose an automatic approach for extracting challenge sets replete with long-distance dependencies, and argue that evaluation using this methodology provides a complementary perspective on system performance. To support our claim, we compile challenge sets for English-German and German-English, which are much larger than any previously released challenge set for MT. The extracted sets are large enough to allow reliable automatic evaluation, which makes the proposed approach a scalable and practical solution for evaluating MT performance on the long-tail of syntactic phenomena.[1]

1 Introduction

The assumption that proximate source words are more likely to correspond to proximate target words has often been introduced as a bias (henceforth, *locality bias*) into statistical MT systems (Brown et al., 1993; Koehn et al., 2003; Chiang, 2005). While reordering phenomena, abundant for some language pairs, violate this simplifying assumption, it has often proved to be a useful inductive bias in practice, especially when complemented with targeted techniques for addressing non-monotonic translation (e.g., Och, 2002; Chiang, 2005). For example, if an adjective precedes a noun in one language and modifies it syntactically, it is likely that their corresponding words will appear close to each other in the translation — i.e., they may not be immediately adjacent or even in the same order in the translation, but it is unlikely that they will be arbitrarily distant from one another.

In the era of Neural Machine Translation (NMT), such biases are implicitly introduced by the sequential nature of the LSTM architecture (Bahdanau et al., 2015, see §2). The influential Transformer model (Vaswani et al., 2017) replaces the sequential LSTMs with self-attention, which does not seem to possess this bias. We show that the default implementation of the Transformer does retain some bias, but that it can be relieved by using learned positional embeddings (§3).

Long-distance dependencies (LDD) between words and phrases present a long-standing problem for MT (Sennrich, 2016), as they are generally more difficult to detect (indeed, they pose an ongoing challenge for parsing as well (Xu et al., 2009)), and often result in non-monotonic translation if the target differs from the source in terms of its word order and lexicalization patterns. The Transformer's indifference to the absolute position of the tokens raises the question of whether long-distance dependencies are still an open problem.

We address this question by proposing an automatic method to compile challenge sets for evaluating system performance on LDD (§4). We distinguish between two main LDD types: (1) reordering LDD, namely cases where source and target words largely correspond to one another but are ordered differently; (2) lexical LDD, where the way a word or a contiguous expression on the target side is translated is dependent on non-adjacent words on the source side.

We define a methodology for extracting both LDD types. For reordering LDD, we build on Birch (2011), whereas for lexical LDD we compile a list of linguistic phenomena that yield LDD,

[1]Our extracted challenge sets and codebase are found in https://github.com/borgr/auto_challenge_sets.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 291–303
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

and use a dependency parser to find instances of these phenomena in the source side of a parallel corpus. As a test case, we apply this method to construct challenge sets (§4.2) for German-English and English-German. The approach can be easily scaled to other languages for which a good enough parser exists.

Experimenting both with RNN and self-attention NMT architectures, we find that although the latter presents no locality bias, LDD remain challenging. Moreover, lexical LDD become increasingly challenging with their distance, suggesting that syntactic distance remains an important determinant of performance in state-of-the-art (SoTA) NMT.

We conclude that evaluating LDD using targeted challenge sets gives a detailed picture of MT performance, and underscores challenges the field has yet to fully address. As particular types of LDD are not frequent enough to significantly affect coarse-grained measures, such as BLEU (Papineni et al., 2002) or TER (Snover et al., 2006), our evaluation approach provides a complementary perspective on system performance.

2 Related Work

2.1 Long-distance Dependencies in MT

A common architecture for text-to-text generation tasks is the (Bi)LSTM encoder-decoder (Bahdanau et al., 2015). This architecture consists of several LSTM layers for the encoder and the decoder and a thin attention layer connecting them. LSTM is a recurrent network with a state vector it updates. At every step, it discards some of the current and past information and aggregates the rest into the state. Any information about the past comes from this state, which is a learned "summary" of the previous states (cf. Greff et al., 2017). Hence, for information to reach a certain prediction step, it should be stored and then kept throughout the intermediate steps (tokens). While theoretically information could be kept indefinitely (Hochreiter and Schmidhuber, 1997), practical evidence shows that LSTMs performance decreases with the distance between the trigger and the prediction (Linzen et al., 2016; Liu et al., 2018), and that they have difficulties generalizing over sequence lengths (Suzgun et al., 2018).

Despite being affected by absolute distances between syntactically dependent tokens (Linzen et al., 2016), LSTMs tend to learn to a certain extent structural information even without being instructed to do so explicitly (Gulordava et al., 2018). Futrell and Levy (2018) discuss similar linguistic phenomena to what we discuss in §4.2, and show that LSTM encoder-decoder systems handle them better than previous N-gram based systems, despite being profoundly affected by distance.

Transformer (Vaswani et al., 2017) models are also encoder-decoder, but instead of LSTMs, they use self-attention. Self-attention is based on gating all outputs of the previous layer as inputs for the current one; put differently, it aggregates all the input in one step. This approach makes information from all parts of the input sequence equally reachable. While this is not the only architecture with such attributes (van den Oord et al., 2016), we focus on it due to its SoTA results for MT (Lakew et al., 2018). The Transformer's use of self-attention inspired other works in related fields (Devlin et al., 2018), some of which attributed their performance gains to the model's ability to capture long-range context (Müller et al., 2018).

As the Transformer does not aggregate input sequentially, token positions must be represented through other means. For that purpose, the embedding of each input token W is concatenated with an embedding of its position in the source sentence P. While positional embeddings can generally be any vectors, two implementations are commonly used (Tebbifakhr et al., 2018; Guo et al., 2018): learned positional embeddings (learnedPEs; P is randomly initialized), and sine positional embeddings (SinePEs) defined as:

$$P_{(pos,2i)} = sin(pos/10,000^{2i/dim})$$
$$P_{(pos,2i+1)} = cos(pos/10,000^{2i/dim})$$

where dim is the dimension of the embedding. Vaswani et al. (2017) report that they see no benefit in learnedPEs, and hence use SinePEs, which have much fewer parameters.

Most of the dependencies between words are short. Short-distance linguistic dependencies include some of the most common phenomena in language, such as determination, modification by an adjective and compounding. For example, 62% of the dependencies in the standard UD EWT training set (Silveira et al., 2014) are between tokens that are up to one word apart. It stands to reason that the locality bias is useful in these cases. Nevertheless, as system quality improves, rarer, more challenging dependencies become a priority,

and languages present a countless number of long-distance reordering phenomena (Deng and Xue, 2017). One example is subject-verb agreement, where a correct translation requires that the verb is inflected according to the headword of the subject (e.g., in English "dogs that ..., bark", while "a dog that ..., barks"). When translating such cases, a locality bias may impede performance, by biasing the model not to attend to both the subject's head and the main verb (which may be arbitrarily distant), thereby disallowing it to correctly inflect the main verb.

Due to the benefits of the locality bias, it featured prominently in statistical MT, including in the IBM models, where alignments are constrained not to cross too much (Brown et al., 1993), and in predicting probabilities of reorderings (Koehn et al., 2003; Chiang, 2005). Difficulties in handling LDD have motivated the development of syntax-based MT (Yamada and Knight, 2001), that can effectively represent reordering at the phrase level, such as when translating between VSO and SOV languages. However, syntax-based MT models remain limited in their ability to map between arbitrarily different word orders (Sun et al., 2009; Xiong et al., 2012). For example, reorderings that violate the assumption that the trees form contiguous phrases would be difficult for most such models to capture. In the next section (§3) we show that the Transformer, when implemented with learnedPEs, presents no locality bias, and hence can, in principle, learn dependencies between any two positions of the source, and use them at any step during decoding.

2.2 MT Evaluation

With major improvements in system performance, crude assessments of performance are becoming less satisfying, i.e., evaluation metrics do not give an indication on the performance of MT systems on important challenges for the field (Isabelle and Kuhn, 2018). String-similarity metrics against a reference are known to be partial and coarse-grained aspects of the task (Callison-Burch et al., 2006), but are still the common practice in various text generation tasks. However, their opaqueness and difficulty to interpret have led to efforts to improve evaluation measures so that they will better reflect the requirements of the task (Anderson et al., 2016; Sulem et al., 2018; Choshen and Abend, 2018b), and to increased interest in defining more interpretable and telling measures (Lo and Wu, 2011; Hodosh et al., 2013; Birch et al., 2016; Choshen and Abend, 2018a).

A promising path forward is complementing string-similarity evaluation with linguistically meaningful challenge sets. Such sets have the advantage of being interpretable: they test for specific phenomena that are important for humans and are crucial for language understanding. Interpretability also means that evaluation artefacts are more likely to be detected earlier. So far, such challenge sets were constructed for French-English (Isabelle et al., 2017; Isabelle and Kuhn, 2018) and English-Swedish (Ahrenberg, 2018) [2]. Previous challenge sets were compiled by manually searching corpora for specific phenomena of interest (e.g., yes-no questions which are formulated differently in English and French). These corpora are carefully made but are small in size (ten examples per phenomenon), which means that evaluation must be done manually as well.

As our methodology extracts sentences automatically based on parser output, we are able to compile much larger challenge sets, which allows us to apply standard MT measures to each subcorpus corresponding to a specific phenomenon. The methodology is, therefore, more flexible, and can be straightforwardly adapted to accommodate future advances in MT evaluation.

3 Locality in SoTA NMT

In this section we show that encoder-decoder models based on BiLSTM with attention (see §2), do exhibit a locality bias, but that the Transformer, whose encoder is based on self-attention, and in which token position is encoded only through learnedPEs, does not present any such bias.

3.1 Methodology

In order to test whether an NMT system presents a locality bias in a controlled environment, we examine a setting of arbitrary absolute order of the source-side tokens. In this case, systems that are predisposed towards monotonic decoding are likely to present lower performance, while systems that have no predisposition as to the order of the target side tokens relative to the source-side tokens are not expected to show any change in per-

[2] In WMT 2019 English-German phenomena were tested with a new corpus, using both human and automatic evaluation. It is not possible, however, to use this evaluation outside the competition (Avramidis et al., 2019).

formance. In order to create a controlled setting, where source-side token order is arbitrary, we extract fixed length sentences, and apply the same permutation to all of them. We then train systems with the permuted source-side data (and the same target-side data), and compare results to a control condition where no permutation is applied.

Concretely, we experiment on a German-English setting, extracting all sentences of the most common length (18) from the WMT2015 (Bojar et al., 2015) training data. This results in 130,983 sentences, of which we hold out 1,000 sentences for testing. It is comparable in training set size to a low-resource language setting.

We set a fixed permutation $\sigma : [18] \rightarrow [18]$ and train systems on three versions of the training data (settings): (1) REGULAR, to be used for control; (2) PERMUTED source-side, in which we apply σ over all source-side tokens; (3) PERPOSEMB where the positional embeddings of the source-side tokens are permuted;[3] and (4) REVERSED, where tokens are input in a reverse order.

We apply the following permutation, σ, to the source-side tokens:

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\ 11 & 5 & 9 & 15 & 8 & 14 & 10 & 1 & 3 & 16 & 12 & 2 & 0 & 6 & 17 & 4 & 13 & 7 \end{pmatrix}$$

We did not find any property that would deem this permutation special (examining, e.g., its decomposition into cycles). We therefore assume that similar results will hold for other σs as well.

We train a Transformer model, optimizing using Adam (Kingma and Ba, 2015). We set the embedding size to 512, dropout rate of 0.1, 6 stack layers in both the encoder and the decoder and 8 attention heads. We use tokenization, truecasing and BPE (Sennrich et al., 2016) as preprocessing, following the same protocol as (Yang et al., 2018).

We experiment both with learnedPEs, and with SinePEs. We train the BiLSTM model using the Nematus implementation (Sennrich et al., 2017b), and use their supplied scripts for preprocessing, training and testing, changing only the datasets used. For all models, we report the highest BLEU score on the test data for any epoch during training, and perform early stopping after 10 consecutive epochs without improvement.

In the Transformer with learnedPEs, 5 repetitions were done in the REGULAR setting, and 5 for the other settings: 5 repetitions for PERMUTED, 1 for PERPOSEMB and 1 for REVERSED. In addition, we trained the BiLSTM model and the Transformer with SinePEs both in the REGULAR condition and in PERMUTED, each was trained once.

[3] Formally, if the source sentence is $(t_1, ..., t_{18})$, then the input to the Transformer is $\left(\left[W(t_1); P(t_{\sigma(1)})\right], ..., \left[W(t_{18}); P(t_{\sigma(18)})\right]\right)$.

Model	Positional	Setting	BLEU
Transformer	LearntPE	REGULAR	24.81
	LearntPE	PERMUTED	24.87 (+0.06)
	LearntPE	REVERSE	24.84 (+0.03)
	LearntPE	PERPOSEMB	24.82 (+0.01)
Transformer	SinePEs	REGULAR	25.08
	SinePEs	PERMUTED	23.90 (-1.18)
Nematus		REGULAR	22.32
		PERMUTED	19.67 (-2.65)

Table 1: BLEU score for various Transformer settings on regular and permuted data. In brackets are the differences from REGULAR. Nematus and Transformer using SinePEs show decreased performance when permuting the input. Transformer with learnedPEs does not. Rows correspond to the different models used (Model), which positional embeddings are fed to the Transformer (Positional), and the order of the input tokens (Setting; see text).

3.2 Results

Table 1 presents our results. We find that Nematus BiLSTM suffers substantially from permuting the source-side tokens, but that the Transformer does not exhibit a locality bias. Indeed, for learnedPEs in all settings (REGULAR, PERMUTED, REVERSED and PERPOSEMB), BLEU scores are essentially the same. We also find that the common practice of using fixed SinePEs does introduce some bias, as attested by the small performance drop between REGULAR and PERMUTED. Like Vaswani et al. (2017), we find that in the REGULAR settings, learnedPEs are not superior in performance to SinePEs, despite having more expressive power. However, our results suggest that the decision between learnedPEs and SinePEs is not without consequences: learnedPEs are preferable if a locality bias is undesired (this is potentially the case for highly divergent language pairs).

3.3 Discussion

Finding that Transformers do not present a locality bias has implications on how to construct their input in MT settings, as well as in other tasks that use self-attention encoders, such as image captioning (You et al., 2016). It is common practice to augment the source-side with globally-applicable

information, e.g., the target language in multilingual MT (Johnson et al., 2017). Having no locality bias implies this additional information can be added at any fixed point in the sequence fed to a Transformer, provided that the positional embeddings do not themselves introduce such a bias. This is not the case with BiLSTMs, which often require introducing the same information at each input token to allow them to be effectively used by the system (Yao et al., 2017; Rennie et al., 2017).

4 LDD Challenge Sets

One of the stated motivations of the Transformer model is to effectively tackle long-distance dependencies, which are "a key challenge in many sequence transduction tasks" (Vaswani et al., 2017). Our results from the previous section show that indeed fixed reordering patterns are completely transparent for Transformers. This, however, still leaves the question of how Transformers handle linguistic reordering patterns, which may involve varying distances between dependent tokens.

4.1 Methodology

We propose a method for scalably compiling challenge sets to support fine-grained MT evaluation for different types of LDD. We address two main types:

Reordering LDD are cases where the words on the two sides of the parallel corpus largely correspond to one another, but are ordered differently. These cases may require attending to source words in a highly non-monotonic order, but the generation of each target word is localized to a specific region in the source sentence. For example, in English-German, the verb in a subordinated clause appears in a final position, while the verb in the English source appears right after the subject. Consider "The man that is sitting on the chair", and the corresponding German "Der Mann, der auf dem Stuhl sitzt" (lit. *the man, that on the chair sits*) — while the verb is placed at different clause positions in the two cases, the words mostly have direct correspondents. Our methodology follows Birch (2011) in detecting such phenomena based on alignment. Concretely, we extract a word alignment between corresponding sentences, and collect all sentences that include a pair of aligned words in the source and target sides, whose indices have a difference of at least $d \in \mathbb{N}$.

Lexical LDD are cases where the translation of a single word or phrase is determined by non-adjacent words on the source side. This requires attending to two or more regions that can be arbitrarily distant from one another. Several phenomena, such as light verbs (Isabelle and Kuhn, 2018), are known from the linguistic and MT literature to yield lexical LDD. Our methodology takes a predefined set of such phenomena, and defines rules for detecting each of them over dependency parses of the source-side. See §4.2 for the list of phenomena we experiment on in this paper.

Focusing on LDD, we restrict ourselves to instances where the absolute distance between the word and the dependent is at least $d \in \mathbb{N}$. Selecting large enough d entails that the extracted phenomena are unlikely to be memorized as a phrase with a specific meaning (e.g., encode "*make* the whole thing *up*" [$d = 3$] as a phrase, rather than as a discontiguous phrase "make ... up" with an argument "the whole thing"). This increases the probability that such cases, if translated correctly, reflect the MT systems' ability to recognize that such discontiguous units are likely to be translated as a single piece.

We note, that by extracting the challenge set based on syntactic parses, we by no means assume these representations are internally represented by the MT systems in any way, or assume such a representation is required for succeeding in correctly translating such constructions. The extraction method is merely a way of finding phenomena we have a reason to believe are difficult to translate, and meaningful for language understanding. We use Universal Dependencies (UD; Nivre et al., 2016) as a syntactic representation, due to its cross-lingual consistency (about 90 languages are supported so far), which allows research on difficult LDD phenomena that recur across languages.

Our extraction methods resemble previous challenge set approaches (Isabelle et al., 2017; Isabelle and Kuhn, 2018; Ahrenberg, 2018), in using linguistically motivated sets of sentence pairs to assess translation quality. However, as our extraction method is fully automatic, it allows for the compilation of much larger challenge sets over many language pairs. The challenge sets we extract contain hundreds or thousands of pairs (§4.2). The size of the sets allows using any MT evaluation measures to measure performance, and is thus a much more scalable solution than manual inspection, as

is commonly done in challenge set approaches.

On the other hand, an automatic methodology has the side-effect of being noisier, and not necessarily selecting the most representative sentences for each phenomenon. For instance *befinden sich* (lit. *to determine*) includes a verb and a reflexive pronoun, which do not necessarily appear contiguously in German. However, as *befinden* always appears with the reflexive *sich*, it might not pose a challenge to NMT systems, which can essentially ignore the reflexive pronoun upon translation.

4.2 A Test Case on Extracting Sets

Next, we discuss the compilation of German-English and English-German corpora. We select these pairs, as they are among the most studied in MT, and comparatively high results are obtained for them (Bojar et al., 2017). Hence, they are more likely to benefit from a fine-grained analysis.

For the reordering LDD corpus, we align each source and target sentences using FastAlign (Dyer et al., 2013) and collect all sentences with at least one pair of source-side and target-side tokens, whose indices have a difference of at least $d = 5$. For example:

> **Source:** *Wäre es ein großer Misserfolg, nicht den Titel in der Ligue 1 zu gewinnen, wie dies in der letzten Saison der Fall war?*
> **Gloss:** *Would-be it a big failure, not the title in the Ligue 1 to win, as this in the last season the case was?*
> **Target:** *In Ligue 1, would not winning the title, like last season, be a big failure?*

We extract lexical LDD using simple rules over source-side parse trees, parsed with UDPipe (Straka and Straková, 2017). For a sentence to be selected, at least one word should separate the detected pair of words. We picked several well-known challenging constructions for translation that involve discontiguous phrases: reflexive-verb, verb-particle constructions and preposition stranding. We note that while these constructions often yield lexical LDDs, and are thus expected to be challenging on average, some of their instances can be translated literally (e.g., *amuse oneself* is translated to *amüsieren sich*).

Reflexive Verbs. Prototypically, reflexivity is the case where the subject and object corefer. Reflexive pronouns in English end with *self* or *selves* (e.g., yourselves) and in German include *sich*, *dich*, *mich* and *uns* among others. However, reflexive pronouns can often change the meaning of a verb unpredictably, and may thus lead to different translations for non-reflexive instances of a verb, compared to reflexive ones. For example, *abheben* in German means taking off (as of a plane), but *sich abheben* means standing out. Similarly, in the example below, *drängte sich* translates to *intrude*, while *drängte* normally translates to *pushed*.

A source sentence is said to include a reflexive verb if one of its tokens is parsed with a reflexive morphological feature (`refl=yes`).
For example:

> **Source:** *[...] es ertragen zu müssen, daß eine unsympathische Fremde* ***sich*** *unaufhörlich in ihren Familienkreis* ***drängte****.*
> **Target:** *[...] to see an uncongenial alien permanently* ***intruded*** *on her own family group.*

Phrasal Verbs are verbs that are made up of a verb and a particle (or several particles), which may change the meaning of the verb unpredictably. Examples of English phrasal verbs include *run into* (in the sense of *meet*) and *give in*, and in German they include examples such as *einladen* (*invite*), consisting morphologically of the particle *ein* and the verb *laden* (*load*).

A source sentence is said to include a phrasal verb if a particle dependent (UD labels of `compound:prt` or `prt`) exists in the parse. *trat* in itself means *stepped*, but in the extracted example below, *trat... entgegen* translates to *received*.
For example:

> **Source:** *[...] ich* **trat** *ihm in wahnsinniger Wut* **entgegen**.
> **Target:** *[...] I* **received** *him in frantic sort.*

Preposition Stranding is the case where a preposition does not appear adjacent to the object it refers to. In English, it will often appear at the end of the sentence or a clause. For example, *The banana she stepped* ***on*** or *The boy I read the book* ***to***. Preposition stranding is common in English and other languages such as Scandinavian languages or Dutch (Hornstein and Weinberg, 1981). However, in German, it is not a part of standard written language (Beermann and Ik-Han, 2005), although it does (rarely) appear (Fanselow, 1983). We, therefore, extract this challenge set only with English as the source side.

While preposition stranding is often regarded as a syntactic phenomenon, we consider it here a lexical LDD, since the translation of prepositions

	Phenomena	Books	Newstest2013
De↔En	Reorder	7,457	306
	Baseline (full dataset)	51,467	3,000

Table 2: Sizes of reordering and baseline corpora.

		Min Distance				
	Phenomena	All	≥1	≥2	≥3	News
De→En	Particle	8,361	7,584	6,261	4,780	232
	Reflexive	13,207	8,122	5,598	4,226	281
En→De	Particle	4,636	786	111	36	17
	Reflexive	3,225	1,188	460	274	11
	Preposition Stranding	682	191	85	40	8

Table 3: Sizes of Lexical LDD corpora. Challenge sets are partitioned (in order of appearance) by the language pairs, the phenomenon type, and the minimal distance between the head and the dependent. Phenomenon appears in the source. Statistics for the Newstest2013 corpora with miminal distance ≥ 1 are at the rightmost column, the rest are on Books.

(and in some cases their accompanying verbs) is dependent on the prepositional object, which in the case of preposition stranding, may be distant from the preposition itself. For example, translating *the* **car** *we looked* **for** into German usually uses the verb *suchen* (*search*), while translating *the* **car** *we looked* **at** does not. Translating prepositions is difficult in general (Hashemi and Hwa, 2014), but preposition stranding is especially so, as there is no adjacent object to assist disambiguation.

A source sentence is said to include preposition stranding if it contains two nodes with an edge of the type `obl` (oblique) or a subcategory thereof between them, and the UD POS tag of the dependent is adposition (ADP).

For example,

Source: *[...] wherever she wanted to* **send** *the hedgehog* **to** *[...]*
Gloss: *[...] where she the hedgehog rolled-towards wanted [...]*
Target: *[...] wo sie den Igel hinrollen wollte [...]*

4.3 Experiments

We turn to evaluate SoTA NMT performance on the extracted challenge sets.

Experimental Setup. We trained the Transformer on WMT2015 training data (Bojar et al., 2015), for parameters see §3.1. For Nematus we used the non-ensemble pre-trained model from (Sennrich et al., 2017a). Each of the test sets, either a baseline or a challenge sets, for the Transformer and Nematus used a maximum of 10k and 1k sentences per set respectively.[4]

		Transformer		Nematus	
		Books	News	Books	News
De→En	Baseline	9.02	28.23	16.26	26.32
	Reorder	7.16	22.68	13.88	22.73
	Particle	7.52	27.46	15.41	23.98
	Reflexive	8.15	27.84	14.91	27.04
En→De	Baseline	6.33	23.7	12.25	22.03
	Reorder	4.31	19.4	9.02	20.38
	Particle	5.30	17.83	9.55	16.72
	Reflexive	5.07	15.77	9.97	21.81
	Preposition Stranding	5.37	11.82	9.73	6.27

Table 4: BLEU scores on the challenge sets. Minimum distance between head and dependent $d \geq 1$. A clear, consistent drop from the Baseline (full corpus) score is observed in all cases. The top part of the table corresponds to German-to-English (De→En) sets, and bottom part to English-to-German (En→De) sets. Within each part, rows correspond to various linguistic phenomena (second column), including reordering LDD (Reorder), Verb-Particle Constructions (Particle), Reflexive Verbs (Reflexive) and Preposition Stranding. Columns correspond to the models (Tranformer/Nematus), and the domains (Books/News).

Two parallel corpora were used for extracting the challenge sets. One is newstest2013 (Bojar et al., 2015) from the news domain that is commonly used as a development set for English-German. The other is the relatively unused Books corpus (Tiedemann, 2012) from the more challenging domain of literary translation. The corpora are of sizes 51K and 3K respectively. For lexical LDD, we took the distance (d) between the relevant words to be at least 1, meaning there is at least one word separating them. See Tables 2, 3 for the sizes of the extracted corpora.

For evaluation, we use the MOSES implementation of BLEU (Papineni et al., 2002; Koehn et al., 2007), and for reordering LDD, also RIBES (Isozaki et al., 2010), which focuses on reordering. RIBES measures the correlation of n-gram ranks between the output and the reference, where n-gram appears uniquely and in both.

Manual Validation. To assess the ability of our procedure to extract relevant LDDs, we manually analyzed over 180 source German sentences ex-

[4]We subsample a smaller test set for Nematus, since the most competitive model for the language pair requires Theano. As Theano is deprecated for two years now, it cannot run on our GPUs, which entails long inference time.

	Language	Phenomena	BLEU All	BLEU ≥1	BLEU ≥2	BLEU ≥3	Spearman
Transformer	German	Particle	7.56	7.52	7.5	7.49	-0.96
		Reflexive	8.26	8.15	8.04	-	-1
	English	Particle	4.96	5.3	4.96	6.01	0.73
		Reflexive	5.41	5.07	5.25	5.04	-0.63
		Preposition Stranding	5	5.37	4.42	4.64	-0.63
Nematus	German	Particle	15.48	15.41	14.45	12.36	-0.92
		Reflexive	15.27	14.91	15.13	13.14	-0.80
	English	Particle	10.14	9.55	9.36	8.82	-0.98
		Reflexive	9.46	9.97	9.54	9.35	-0.36
		Preposition Stranding	10.01	9.73	9.14	9.04	-0.97

Table 5: The effect of dependency distance for lexical LDDs on SoTA performance . Results are in BLEU over the Books challenge sets. Columns correspond to the minimum distance, where *All* does not restrict distance (control). The rightmost column presents the Spearman correlation of the phenomena's score with the minimum distance used. All correlations but one are highly negative, implying that distance has a negative effect on performance.

tracted from Books, and 81 English ones including all the instances extracted from News and 45 extracted from Books, where instances are evenly distributed between phenomena and distance of exactly 1,2 or 5. We find that 85% of German sentences, 87% of the English News sentences and 86% of the Books ones indeed contain the target phenomenon. For details of the manual evaluation of the extraction procedure, see Appendix 1.

		News	Books
German	Baseline	0.82	0.57
	Reorder	0.79	0.54
English	Baseline	0.79	0.56
	Reorder	0.77	0.53

Table 6: RIBES scores on the reordering LDD challenge sets. Sentences extracted as being challenging to reorder are harder for the Transformer (lower score). This trend is consistent with our experiments with BLEU. First column indicates the source language.

Results. Comparison of the overall BLEU scores of the NMT models (Table 4) against their performance on the challenge sets, shows that the phenomena are challenging for both models. Both in the small development set of newstest2013 and the large set of Books, the challenge subparts are more challenging across the board. For reordering LDD, we further apply RIBES and find a similar trend: RIBES score is lower for the reorder challenge set than the baseline (see Table 6).

In order to confirm that the distance between the head and dependent (the "length" of the dependency) is related to the observed performance drop in the case of lexical LDD, we partition each of the challenge sets according to their length (d), and compare the results to a control condition, where all instances of the phenomena listed in §4.2 are extracted, including non-LDD instances, i.e., sentences where the head and the dependent are adjacent. System performance on the sliced challenge sets (Table 5) shows that performance indeed decreases with d. Results thus indicate that it is not only the presence of the phenomena that make these sets challenging, but that the challenge increases with the distance.

We validate this main finding using manual annotation of German to English cases. Using two annotators (with high agreement between them; κ=0.79), we find that the decrease in performance with d is replicated. We measure how many of the detected lexical LDD are correctly translated, ignoring the rest of the source and output, as done in manual challenge set approaches. We find that 60%, 54% and 38% of the cases are translated correctly for $d \in (1, 2, 5)$, respectively. This suggests that the extracted phenomena and the distance indeed pose a challenge, and that the automatic metric we use shows the correct trend in these cases. See Appendix 2 for details.

Discussion. Interestingly, these results hold true for the Transformer despite its indifference to the absolute word order. Therefore, word distance in itself is not what makes such phenomena challenging, contrary to what one might expect from the definition of LDD. It seems then that these

phenomena are especially challenging due to the non-standard linguistic structure (e.g., syntactic and lexical structure), and the varying distances in which LDD manifest themselves. The models, therefore, seem to be unable to learn the linguistic structure underlying these phenomena, which may motivate more explicit modelling of linguistic biases into NMT models, as proposed by, e.g., Eriguchi et al. (2017) and Song et al. (2019).

We note that our experiments were not designed to compare the performance of BiLSTM and self-attention models. We, therefore, do not see the Transformer's inferior performance on Books, relative to Nematus as an indication of the general ability of this model in out-of-domain settings. What is evident from the results is that translating Books is a challenge in itself, probably due to the register of the language, and the presence of frequent non-literal translations.

A potential confound is that performance might change with the length of the source in BiLSTMs (Carpuat et al., 2013; Murray and Chiang, 2018), in Transformers it was reported to increase (Zhang et al., 2018). Length is generally greater in the challenge set than in the full test set, and generally increases with d, showing if anything a decrease of performance by length. To assess whether our corpora are challenging due to a length bias, we randomly sample from Books 1,000 corpora with 1,000, 100 and 10 sentences each. The correlation between their corresponding average length and the Transformers' BLEU score on them was 0.06,0.09 and 0.03 respectively. While this suggests length is not a strong predictor of performance, to verify that difficulty is not a result of the distribution of lengths in the challenge sets we conduct another experiment.

For each challenge set and each value of d (0–3), we sample 100 corpora. For each sentence in a given challenge set, we sample a sentence of no more than a difference of 1 in length. This results in a corpus with a similar length distribution, but sampled from the overall population of Books sentences. Results show that the BLEU score of the challenge sets in all German to English cases is lower than any randomly sampled corpus.[5] In the English-German cases, trends are similar, albeit less pronounced. This may be due to the low number of long English sentences, which lead to more homogeneous samples. Overall, results suggest that length is extremely unlikely to be the only cause for the observed trends.

[5]Most sampled corpora actually had better scores than the baseline. We believe this is because very short sentences which are mostly noise, are never sampled.

5 Conclusion

As NMT system performance is constantly improving, more reliable methods for identifying and classifying their failures are needed. Much research effort is therefore devoted to developing more fine-grained and interpretable evaluation methods, including challenge-set approaches. In this paper, we showed that, using a UD parser, it is possible to extract challenge sets that are large enough to allow scalable MT evaluation of important and challenging phenomena.

An accumulating body of research is devoted to the ability of modern neural architectures such as LSTMs (Linzen et al., 2016) and pretrained embeddings (Hewitt and Manning, 2019; Liu et al., 2019; Jawahar et al., 2019) to represent linguistic features. This paper makes a contribution to this literature in confirming that the Transformer model can indeed be made indifferent to the absolute order of the words, but also shows that this does not entail that the model can overcome the difficulties of LDD in naturalistic data. We may carefully conclude then that despite the remarkable feats of current NMT models, inducing linguistic structure in its more evasive and challenging instances is still beyond the reach of state-of-the-art NMT, which motivates exploring more linguistically-informed models.

6 Acknowledgments

This work was supported by the Israel Science Foundation (grant no. 929/17)

References

Lars Ahrenberg. 2018. A challenge set for english-swedish machine translation. In *SLTC*.

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer.

Eleftherios Avramidis, Vivien Macketanz, Ursula Strohriegel, and Hans Uszkoreit. 2019. Linguistic evaluation of german-english machine translation using a test suite. In *Proceedings of the Fourth Conference on Machine Translation. Conference on Machine Translation (WMT-2019), located at The*

57th Annual Meeting of the Association for Computational Linguistics, August 1-2, Florence, Italy. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Dorothee Beermann and Lars Ik-Han. 2005. Preposition stranding and locative adverbs in german. *Organizing Grammar*, 86:31.

Alexandra Birch. 2011. *Reordering metrics for statistical machine translation*. Ph.D. thesis, The University of Edinburgh.

Alexandra Birch, Omri Abend, Ondřej Bojar, and Barry Haddow. 2016. Hume: Human ucca-based evaluation of machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1264–1274.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. 2017. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on statistical machine translation. In *WMT@EMNLP*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluation the role of bleu in machine translation research. In *EACL*.

Marine Carpuat, Lucia Specia, and Dekai Wu. 2013. Proceedings of ssst-7, seventh workshop on syntax, semantics and structure in statistical translation. In *EMNLP 2014*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270. Association for Computational Linguistics.

Leshem Choshen and Omri Abend. 2018a. Automatic metric validation for grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Leshem Choshen and Omri Abend. 2018b. Reference-less measure of faithfulness for grammatical error correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Dun Deng and Nianwen Xue. 2017. Translation divergences in chinese–english machine translation: An empirical investigation. *Computational Linguistics*, 43(3):521–565.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *ACL*.

Gisbert Fanselow. 1983. Zu einigen problemen von kasus, rektion und bindung in der deutschen syntax. *Universität Konstanz*.

Richard Futrell and Roger P. Levy. 2018. Do rnns learn human-like abstract word order preferences? *CoRR*, abs/1811.01866.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2017. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *NAACL-HLT*.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2018. Star-transformer. *arXiv preprint arXiv:1902.09113*.

Homa B Hashemi and Rebecca Hwa. 2014. A comparison of mt errors and esl errors. In *LREC*, pages 2696–2700.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Norbert Hornstein and Amy Weinberg. 1981. Case theory and preposition stranding. *Linguistic inquiry*, 12(1):55–91.

Pierre Isabelle, Colin Cherry, and George F. Foster. 2017. A challenge set approach to evaluating machine translation. In *EMNLP*.

Pierre Isabelle and Roland Kuhn. 2018. A challenge set for french–> english machine translation. *arXiv preprint arXiv:1806.02725*.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Surafel Melaku Lakew, Mauro Cettolo, and Marcello Federico. 2018. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. In *COLING*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Nelson F. Liu, Omer Levy, Roy Schwartz, Chenhao Tan, and Noah A. Smith. 2018. Lstms exploit linguistic attributes of data. In *Rep4NLP@ACL*.

Chi-kiu Lo and Dekai Wu. 2011. Meant: an inexpensive, high-accuracy, semi-automatic metric for evaluating translation utility via semantic frames. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 220–229. Association for Computational Linguistics.

Mathias Müller, Annette Rios, Elena Voita, and Rico Sennrich. 2018. A large-scale test set for the evaluation of context-aware pronoun translation in neural machine translation. In *WMT*.

Kenton Murray and David Chiang. 2018. Correcting length bias in neural machine translation. In *WMT*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proc. of LREC*, pages 1659–1666.

Franz Josef Och. 2002. *Statistical machine translation: from single-word models to alignment templates*. Ph.D. thesis, Bibliothek der RWTH Aachen.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. In *SSW*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

Rico Sennrich. 2016. How grammatical is characterlevel neural machine translation. *Assessing MT quality with contrastive translation pairs. CoRR, abs/1612.04629*.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017a. The university of edinburgh's neural mt systems for wmt17. In *WMT*.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017b. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1715–1725.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Linfeng Song, Daniel Gildea, Yue Zhang, Zhiguo Wang, and Jinsong Su. 2019. Semantic neural machine translation using amr. *arXiv preprint arXiv:1902.07282*.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018. Semantic structural evaluation for text simplification. In *NAACL-HLT*.

Jun Sun, Min Zhang, and Chew Lim Tan. 2009. A non-contiguous tree sequence alignment-based model for statistical machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 914–922. Association for Computational Linguistics.

Mirac Suzgun, Yonatan Belinkov, and Stuart M Shieber. 2018. On evaluating the generalization of lstm models in formal languages. *arXiv preprint arXiv:1811.01001*.

Amirhossein Tebbifakhr, Ruchit Agrawal, Matteo Negri, and Marco Turchi. 2018. Multi-source transformer with combined losses for automatic post editing. In *WMT*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Lrec*, volume 2012, pages 2214–2218.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for smt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 902–911. Association for Computational Linguistics.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve smt for subject-object-verb languages. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 245–253. Association for Computational Linguistics.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.

Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2018. Improving neural machine translation with conditional sequence generative adversarial nets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1346–1355. Association for Computational Linguistics.

Ting Yao, Yingwei Pan, Yehao Li, Zhaofan Qiu, and Tao Mei. 2017. Boosting image captioning with attributes. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4904–4912.

Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4651–4659.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. In *ACL*.

Low-Resource Parsing with Crosslingual Contextualized Representations

Phoebe Mulcaire[♡*] **Jungo Kasai**[♡*] **Noah A. Smith**[♡♢]
[♡]Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA, USA
[♢]Allen Institute for Artificial Intelligence, Seattle, WA, USA
{pmulc,jkasai,nasmith}@cs.washington.edu

Abstract

Despite advances in dependency parsing, languages with small treebanks still present challenges. We assess recent approaches to multilingual contextual word representations (CWRs), and compare them for crosslingual transfer from a language with a large treebank to a language with a small or nonexistent treebank, by sharing parameters between languages in the parser itself. We experiment with a diverse selection of languages in both simulated and truly low-resource scenarios, and show that multilingual CWRs greatly facilitate low-resource dependency parsing even without crosslingual supervision such as dictionaries or parallel text. Furthermore, we examine the non-contextual part of the learned language models (which we call a "decontextual probe") to demonstrate that polyglot language models better encode crosslingual lexical correspondence compared to aligned monolingual language models. This analysis provides further evidence that polyglot training is an effective approach to crosslingual transfer.

1 Introduction

Dependency parsing has achieved new states of the art using distributed word representations in neural networks, trained with large amounts of annotated data (Dozat and Manning, 2017; Dozat et al., 2017; Ma et al., 2018; Che et al., 2018). However, many languages are low-resource, with small or no treebanks, which presents a severe challenge in developing accurate parsing systems in those languages. One way to address this problem is with a crosslingual solution that makes use of a language with a large treebank and raw text in both languages. The hypothesis behind this approach is that, although each language is unique, different languages manifest similar characteristics (e.g., morphological, lexical, syntactic) which can be exploited by training a single *polyglot* model with data from multiple languages (Ammar, 2016).

Recent work has extended contextual word representations (CWRs) multilingually either by training a polyglot language model (LM) on a mixture of data from multiple languages (*joint training* approach; Mulcaire et al., 2019; Lample and Conneau, 2019) or by aligning multiple monolingual language models crosslingually (*retrofitting* approach; Schuster et al., 2019; Aldarmaki and Diab, 2019). These multilingual representations have been shown to facilitate crosslingual transfer on several tasks, including Universal Dependencies parsing and natural language inference. In this work, we assess these two types of methods by using them for low-resource dependency parsing, and discover that the joint training approach substantially outperforms the retrofitting approach. We further apply multilingual CWRs produced by the joint training approach to diverse languages, and show that it is still effective in transfer between distant languages, though we find that phylogenetically related source languages are generally more helpful.

We hypothesize that joint polyglot training is more successful than retrofitting because it induces a degree of lexical correspondence between languages that the linear transformation used in retrofitting methods cannot capture. To test this hypothesis, we design a *decontextual probe*. We *decontextualize* CWRs into non-contextual word vectors that retain much of CWRs' task-performance benefit, and evaluate the crosslingual transferability of language models via word translation. In our decontextualization framework, we use a single LSTM cell without recurrence to obtain a context-independent vector, thereby allowing for a direct probe into the LSTM networks in-

[*] Equal contribution. Random order.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 304–315
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

dependent of a particular corpus. We show that decontextualized vectors from the joint training approach yield representations that score higher on a word translation task than the retrofitting approach or word type vectors such as fastText (Bojanowski et al., 2017). This finding provides evidence that polyglot language models encode crosslingual similarity, specifically crosslingual lexical correspondence, that a linear alignment between monolingual language models does not.

2 Models

We examine crosslingual solutions to low-resource dependency parsing, which make crucial use of multilingual CWRs. All models are implemented in AllenNLP, version 0.7.2 (Gardner et al., 2018) and the hyperparameters and training details are given in the appendix.

2.1 Multilingual CWRs

Prior methods to produce multilingual contextual word representations (CWRs) can be categorized into two major classes, which we call *joint training* and *retrofitting*.[1] The joint training approach trains a single polgylot language model (LM) on a mixture of texts in multiple languages (Mulcaire et al., 2019; Lample and Conneau, 2019; Devlin et al., 2019),[2] while the retrofitting approach trains separate LMs on each language and aligns the learned representations later (Schuster et al., 2019; Aldarmaki and Diab, 2019). We compare example approaches from these two classes using the same LM training data, and discover that the joint training approach generally yields better performance in low-resource dependency parsing, even without crosslingual supervision.

Retrofitting Approach Following Schuster et al. (2019), we first train a bidirectional LM with two-layer LSTMs on top of character CNNs for each language (ELMo, Peters et al., 2018), and then align the monolingual LMs across languages. Denote the hidden state in the jth layer for word i in context c by $\mathbf{h}_{i,c}^{(j)}$. We use a trainable weighted average of the three layers (character-CNN and two LSTM layers) to compute the contextual representation $\mathbf{e}_{i,c}$ for the word: $\mathbf{e}_{i,c} = \sum_{j=0}^{2} \lambda_j \mathbf{h}_{i,c}^{(j)}$ (Peters et al., 2018).[3] In the first step, we compute an "anchor" $\mathbf{h}_i^{(j)}$ for each word by averaging $\mathbf{h}_{i,c}^{(j)}$ over all occurrences in an LM corpus. We then apply a standard dictionary-based technique[4] to create multilingual word embeddings (Mikolov et al., 2013; Conneau et al., 2018). In particular, suppose that we have a word-translation dictionary from source language s to target language t. Let $\mathbf{H}_s^{(j)}, \mathbf{H}_t^{(j)}$ be matrices whose columns are the anchors in the jth layer for the source and corresponding target words in the dictionary. For each layer j, find the linear transformation $\mathbf{W}^{*(j)}$ such that

$$\mathbf{W}^{*(j)} = \underset{\mathbf{W}}{\operatorname{argmin}} ||\mathbf{W}\mathbf{H}_s^{(j)} - \mathbf{H}_t^{(j)}||_F$$

The linear transformations are then used to map the LM hidden states for the source language to the target LM space. Specifically, contextual representations for the source and target languages are computed by $\sum_{j=0}^{2} \lambda_j \mathbf{W}^{*(j)} \mathbf{h}_{i,c}^{(j)}$ and $\sum_{j=0}^{2} \lambda_j \mathbf{h}_{i,c}^{(j)}$ respectively. We use publicly available dictionaries from Conneau et al. (2018)[5] and align all languages to the English LM space, again following Schuster et al. (2019).

Joint Training Approach Another approach to multilingual CWRs is to train a single LM on multiple languages (Tsvetkov et al., 2016; Ragni et al., 2016; Östling and Tiedemann, 2017). We train a single bidirectional LM with charater CNNs and two-layer LSTMs on multiple languages (Rosita, Mulcaire et al., 2019). We then use the polyglot LM to provide contextual representations. Similarly to the retrofitting approach, we represent word i in context c as a trainable weighted average of the hidden states in the trained polyglot LM: $\sum_{j=0}^{2} \lambda_j \mathbf{h}_{i,c}^{(j)}$. In contrast to retrofitting, crosslinguality is learned implicitly by sharing all network parameters during LM training; no crosslingual dictionaries are used.

[1] This term was originally used by Faruqui et al. (2015) to describe updates to word vectors, after estimating them from corpora, using semantic lexicons. We generalize it to capture the notion of a separate update to fit something other than the original data, applied after conventional training.

[2] Multilingual BERT is documented in https://github.com/google-research/bert/blob/master/multilingual.md.

[3] Schuster et al. (2019) only used the first LSTM layer, but we found a performance benefit from using all layers in preliminary results.

[4] Conneau et al. (2018) developed an unsupervised alignment technique that does not require a dictionary. We found that their unsupervised alignment yielded substantially degraded performance in downstream parsing in line with the findings of Schuster et al. (2019).

[5] https://github.com/facebookresearch/MUSE#ground-truth-bilingual-dictionaries

Refinement after Joint Training It is possible to combine the two approaches above; the alignment procedure used in the retrofitting approach can serve as a refinement step on top of an already-polyglot language model. We will see only a limited gain in parsing performance from this refinement in our experiments, suggesting that polyglot LMs are already producing high-quality multilingual CWRs even without crosslingual dictionary supervision.

FastText Baseline We also compare the multilingual CWRs to a subword-based, non-contextual word embedding baseline. We train 300-dimensional word vectors on the same LM data using the fastText method (Bojanowski et al., 2017), and use the same bilingual dictionaries to align them (Conneau et al., 2018).

2.2 Dependency Parsers

We train polyglot parsers for multiple languages (Ammar et al., 2016) on top of multilingual CWRs. All parser parameters are shared between the source and target languages. Ammar et al. (2016) suggest that sharing parameters between languages can alleviate the low-resource problem in syntactic parsing, but their experiments are limited to (relatively similar) European languages. Mulcaire et al. (2019) also include experiments with dependency parsing using polyglot contextual representations between two language pairs (English/Chinese and English/Arabic), but focus on high-resource tasks. Here we explore a wider range of languages, and analyze the particular efficacy of a crosslingual approach to dependency parsing in a low-resource setting.

We use a strong graph-based dependency parser with BiLSTM and biaffine attention (Dozat and Manning, 2017), which is also used in related work (Schuster et al., 2019; Mulcaire et al., 2019). Crucially, our parser only takes as input word representations. Universal parts of speech have been shown useful for low-resource dependency parsing (Duong et al., 2015; Ammar et al., 2016; Ahmad et al., 2019), but many realistic low-resource scenarios lack reliable part-of-speech taggers; here, we do not use parts of speech as input, and thus avoid the error-prone part-of-speech tagging pipeline. For the fastText baseline, word embeddings are not updated during training, to preserve crosslingual alignment (Ammar et al., 2016).

Lang	Code	Genus	WALS 81A	Size
English	ENG	Germanic	SVO	–
Arabic	ARA	Semitic	VSO/SVO	5241
Hebrew	HEB	Semitic	SVO	
Croatian	HRV	Slavic	SVO	6983
Russian	RUS	Slavic	SVO	
Dutch	NLD	Germanic	SOV/SVO	12269
German	DEU	Germanic	SOV/SVO	
Spanish	SPA	Romance	SVO	12543
Italian	ITA	Romance	SVO	
Chinese	CMN	Chinese	SVO	3997
Japanese	JPN	Japanese	SOV	
Hungarian	HUN	Ugric	SOV/SVO	910
Finnish	FIN	Finnic	SVO	12217
Vietnamese	VIE	Viet-Muong	SVO	1400
Uyghur	UIG	Turkic	SOV	1656
Kazakh	KAZ	Turkic	SOV	31
Turkish	TUR	Turkic	SOV	3685

Table 1: List of the languages used in our UD v2.2 experiments. Each shaded/unshaded section corresponds to a pair of "related" languages. WALS 81A denotes Feature 81A in WALS, Order of Subject, Object, and Verb (Dryer and Haspelmath, 2013). "Size" represents the downsampled size in # of sentences used for source treebanks. The four languages in bold face are truly low resource languages ($<$ 2000 trees).

3 Experiments

We first conduct a set of experiments to assess the efficacy of multilingual CWRs for low-resource dependency parsing.

3.1 Zero-Target Dependency Parsing

Following prior work on low-resource dependency parsing and crosslingual transfer (Zhang and Barzilay, 2015; Guo et al., 2015; Ammar et al., 2016; Schuster et al., 2019), we conduct multi-source experiments on six languages (German, Spanish, French, Italian, Portuguese, and Swedish) from Google universal dependency treebank version 2.0 (McDonald et al., 2013).[6] We train language models on the six languages and English to produce multilingual CWRs. For each tested language, we train a polyglot parser with the multilingual CWRs on the five other languages and English, and apply the parser to the test data for the target language. Importantly, the parsing annotation scheme is shared among the seven languages. Our results will show that the joint training approach for CWRs substantially outperforms the retrofitting approach.

[6] http://github.com/ryanmcd/uni-dep-tb

3.2 Diverse Low-Resource Parsing

The previous experiment compares the joint training and retrofitting approaches in low-resource dependency parsing only for relatively similar languages. In order to study the effectiveness more extensively, we apply it to a more typologically diverse set of languages. We use five pairs of languages for "low-resource simulations," in which we reduce the size of a large treebank, and four languages for "true low-resource experiments," where only small UD treebanks are available, allowing us to compare to other work in the low-resource condition (Table 1). Following de Lhoneux et al. (2018), we selected these language pairs to represent linguistic diversity. For each target language, we produce multilingual CWRs by training a polyglot language model with its related language (e.g., Arabic and Hebrew) as well as English (e.g., Arabic and English). We then train a polyglot dependency parser on each language pair and assess the crosslingual transfer in terms of target parsing accuracy.

Each pair of related languages shares features like word order, morphology, or script. For example, Arabic and Hebrew are similar in their rich transfixing morphology (de Lhoneux et al., 2018), and Dutch and German share most of their word order features. We chose Chinese and Japanese as an example of a language pair which does *not* share a language family but does share characters.

We chose Hungarian, Vietnamese, Uyghur, and Kazakh as true low-resource target languages because they had comparatively small amounts of annotated text in the UD corpus (Vietnamese: 1,400 sentences, 20,285 tokens; Hungarian: 910 sentences, 20,166 tokens; Uyghur: 1,656 sentences, 19,262 tokens; Kazakh: 31 sentences, 529 tokens;), yet had convenient sources of text for LM pretraining (Zeman et al., 2018).[7] Other small treebanks exist, but in most cases another larger treebank exists for the same language, making domain adaptation a more likely option than crosslingual transfer. Also, recent work (Che et al., 2018) using contextual embeddings was top-ranked for most of these languages in the CoNLL 2018 shared task on UD parsing (Zeman et al., 2018).[8]

We use the same Universal Dependencies (UD) treebanks (Nivre et al., 2018) and train/development/test splits as the CoNLL 2018 shared task (Zeman et al., 2018).[9] The annotation scheme is again shared across languages, which facilitates crosslingual transfer. For each triple of two related languages and English, we downsample training and development data to match the language with the smallest treebank size. This allows for fairer comparisons because within each triple, the source language for any parser will have the same amount of training data. We further downsample sentences from the target train/development data to simulate low-resource scenarios. The ratio of training and development data is kept 5:1 throughout the simulations, and we denote the number of sentences in training data by $|D_\tau|$. For testing, we use the CoNLL 2018 script on the gold word segmentations. For the truly low-resource languages, we also present results with word segmentations from the system outputs of Che et al. (2018) (HUN, VIE, UIG) and Smith et al. (2018) (KAZ) for a direct comparison to those languages' best previously reported parsers.[10]

4 Results and Discussion

In this section we describe the results of the various parsing experiments.

4.1 Zero-Target Parsing

Table 2 shows results on zero-target dependency parsing. First, we see that all CWRs greatly improve upon the fastText baseline. The joint training approach (Rosita), which uses no dictionaries, consistently outperforms the dictionary-dependent retrofitting approach (ELMos+Alignment). As discussed in the previous section, we can apply the alignment method to refine the already-polyglot Rosita using dictionaries. However, we observe a relatively limited gain in overall performance (74.5 vs. 73.9 LAS points), suggesting that Rosita (polyglot language model) is already developing useful multilingual CWRs for parsing without crosslingual supervision. Note that the degraded overall performance of our ELMo+Alignment compared to Schuster et al. (2019)'s reported results (71.2 vs. 73.1) is likely

[7] The one exception is Uyghur where we only have 3M words in the raw LM data from Zeman et al. (2018).

[8] In Kazakh, Che et al. (2018) did not use CWRs due to the extremely small treebank size.

[9] See Appendix for a list of UD treebanks used.

[10] System outputs for all shared task systems are available at https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-2885

Model	DEU	SPA	FRA	ITA	POR	SWE	AVG
Schuster et al. (2019) (retrofitting)	61.4	77.5	**77.0**	77.6	73.9	71.0	73.1
Schuster et al. (2019) (retrofitting, no dictionaries)	**61.7**	76.6	76.3	**77.1**	69.1	54.2	69.2
fastText + Alignment	45.2	68.5	62.8	58.9	61.1	50.4	57.8
ELMos + Alignment (retrofitting)	57.3	75.4	73.7	71.6	75.1	74.2	71.2
Rosita (joint training, no dictionaries)	58.0	**81.8**	75.6	74.8	**77.1**	76.2	73.9
Rosita + Refinement (joint training + retrofitting)	**61.7**	79.7	75.8	76.0	76.8	**76.7**	**74.5**

Table 2: Zero-target results in LAS. Results reported in prior work (above the line) use an unknown amount of LM training data; all models below the line are limited to approximately 50M words per language.

Model	DEU	SPA	FRA	ITA	POR	SWE	AVG
Zhang and Barzilay (2015)	54.1	68.3	68.8	69.4	72.5	62.5	65.9
Guo et al. (2016)	55.9	73.1	71.0	71.2	78.6	69.5	69.9
Ammar et al. (2016)	57.1	74.6	73.9	72.5	77.0	68.1	70.5
Schuster et al. (2019) (retrofitting)	**65.2**	80.0	**80.8**	**79.8**	82.7	75.4	77.3
Schuster et al. (2019) (retrofitting, no dictionaries)	64.1	77.8	79.8	79.7	79.1	69.6	75.0
Rosita (joint training, no dictionaries)	63.6	**83.4**	78.9	77.8	83.0	**79.6**	77.7
Rosita + Refinement (joint training + retrofitting)	64.8	82.1	78.7	78.8	**84.1**	79.1	**77.9**

Table 3: Zero-target results in LAS with gold UPOS.

due to the significantly reduced amount of LM data we used in all of our experiments (50M words per language, an order of magnitude reduction from the full Wikipedia dumps used in Schuster et al. (2019)). Schuster et al. (2019) (no dictionaries) is the same retrofitting approach as ELMos+Alignment except that the transformation matrices are learned in an unsupervised fashion without dictionaries (Conneau et al., 2018). The absence of a dictionary yields much worse performance (69.2 vs. 73.1) in contrast with the joint training approach of Rosita, which also does not use a dictionary (73.9).

We also present results using gold universal part of speech to compare to previous work in Table 3. We again see Rosita's effectiveness and a marginal benefit from refinement with dictionaries. It should also be noted that the reported results for French, Italian and German in Schuster et al. (2019) outperform all results from our controlled comparison; this may be due to the use of abundant LM training data. Nevertheless, joint training, with or without refinement, performs best on average in both gold and predicted POS settings.

4.2 Diverse Low-Resource Parsing

Low-Resource Simulations Figure 1 shows simulated low-resource results.[11] Of greatest interest are the significant improvements over monolingual parsers when adding English or related-language data. This improvement is consistent across languages and suggests that crosslingual transfer is a viable solution for a wide range of languages, even when (as in our case) language-specific tuning or annotated resources like parallel corpora or bilingual dictionaries are not available. See Figure 2 for a visualization of the differences in performance with varying training size. The polyglot advantage is minor when the target language treebank is large, but dramatic in the condition where the target language has only 100 sentences. The fastText approaches consistently underperform the language model approaches, but show the same pattern.

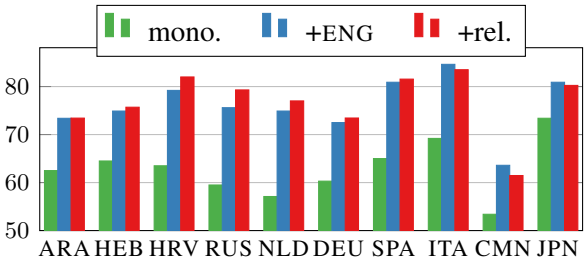


Figure 1: LAS for UD parsing results in a simulated low-resource setting where the size of the target language treebank ($|D_\tau|$) is set to 100 sentences.

In addition, related-language polyglot ("+rel.") outperforms English polyglot in most cases in the low-resource condition. The exceptions to this pattern are Italian (whose treebank is of a different genre from the Spanish one), and Japanese and Chinese, which differ significantly in morphology and word order. The CMN/JPN result suggests that such typological features influence the degree of crosslingual transfer more than orthographic

[11] A table with full details including different size simulations is provided in the appendix.

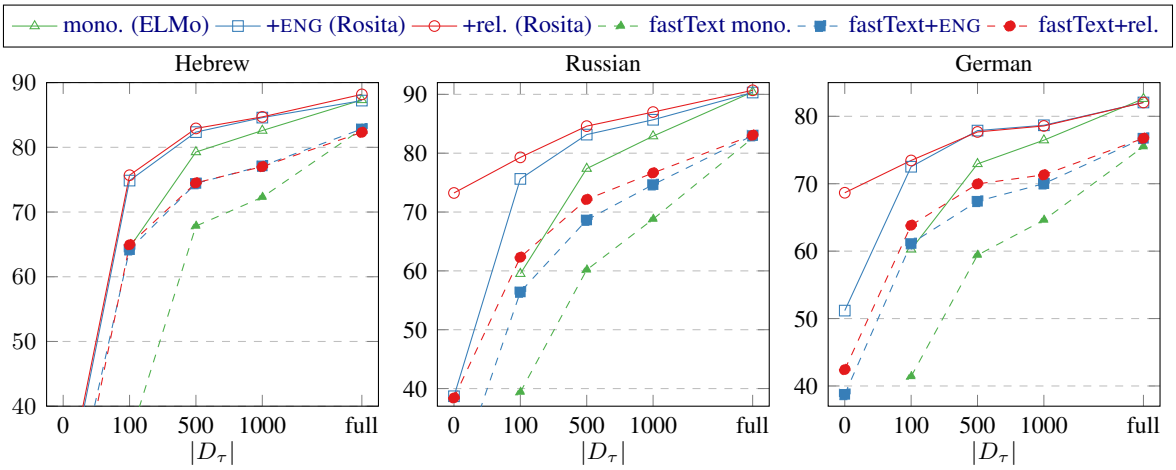


Figure 2: Plots of parsing performance vs. target language treebank size for several example languages. The size 0 target treebank point indicates a parser trained *only* on the source language treebank but with polyglot representations, allowing transfer to the target test treebank using no target language training trees. See Appendix for results with zero-target-treebank and intermediate size data ($|D_\tau| \in \{0, 100, 500, 1000\}$) for all languages.

properties like shared characters. This result in crosslingual transfer also mirrors the observation from prior work (Gerz et al., 2018) that typological features of the language are predictive of *monolingual* LM performance. The related-language improvement also vanishes in the full-data condition (Figure 2), implying that the importance of shared linguistic features can be overcome with sufficient annotated data. It is also noteworthy that variations in word order, such as the order of adjective and noun, do not affect performance: Italian, Arabic, and others use a noun-adjective order while English uses an adjective-noun order, but their +ENG and +rel. results are comparable.

The Croatian and Russian results are notable because of shared heritage but different scripts. Though Croatian uses the Latin alphabet and Russian uses Cyrillic, transfer between HRV+RUS is clearly more effective than HRV+ENG (82.00 vs. 79.21 LAS points when $|D_\tau| = 100$). This suggests that character-based LMs can implicitly learn to transliterate between related languages with different scripts, even without parallel supervision.

Truly Low Resource Languages Finally we present "true low-resource" experiments for four languages in which little UD data is available (see Section 3.2). Table 4 shows these results. Consistent with our simulations, our parsers on top of Rosita (multilingual CWRs from the joint training approach) substantially outperform the parsers with ELMos (monolingual CWRs) in all languages, and establish a new state of the art

Model	gold	pred.
Hungarian (HUN)		
Che et al. (2018) (HUN, ensemble)	–	82.66
Che et al. (2018) (HUN)	–	80.96
ELMo (HUN)	81.89	81.54
Rosita (HUN +ENG)	85.34	84.89
Rosita (HUN +FIN)	**85.40**	**84.96**
Vietnamese (VIE)		
Che et al. (2018) (VIE, ensemble)	–	55.22
ELMo (VIE)	62.67	55.72
Rosita (VIE +ENG)	**63.07**	**56.42**
Uyghur (UIG)		
Che et al. (2018) (UIG, ensemble)	–	**67.05**
Che et al. (2018) (UIG)	–	66.20
ELMo (UIG)	66.64	63.98
Rosita (UIG +ENG)	67.85	65.55
Rosita (UIG +TUR)	**68.08**	65.73
Kazakh (KAZ)		
Rosa and Mareček (2018) (KAZ +TUR)	–	26.31
Smith et al. (2018) (KAZ +TUR)	–	31.93
Schuster et al. (2019) (KAZ +TUR)	–	36.98
Rosita (KAZ +ENG)	48.02	46.03
Rosita (KAZ +TUR)	**53.98**	**51.96**

Table 4: LAS (F_1) comparison for truly low-resource languages. The gold and pred. columns show results under gold segmentation and predicted segmentation. The languages in the parentheses indicate the languages used in parser training.

in Hungarian, Vietnamese, and Kazakh. Consistent with our simulations, we see that training parsers with the target's related language is more effective than with the more distant language, English. It is particularly noteworthy that the Rosita models, which do not use a parallel corpus or dictionary, dramatically improve over the best previously reported result from Schuster et al. (2019) when either the related language of Turkish (51.96 vs. 36.98) or even the more distant language of English (46.03 v.s. 36.98) is used. Schuster et al. (2019) aligned the monolingual ELMos for Kazakh and Turkish using the KAZ-TUR dictionary that Rosa and Mareček (2018) derived from parallel text. This result further corroborates our finding that the joint training approach to multilingual CWRs is more effective than retrofitting monolingual LMs.

4.3 Comparison to Multilingual BERT Embeddings

We also evaluate the diverse low-resource language pairs using pretrained multilingual BERT (Devlin et al., 2019) as text embeddings (Figure 3). Here, the same language model (multilingual cased BERT,[12] covering 104 languages) is used for all parsers, with the only variation being in the training treebanks provided to each parser. Parsers are trained using the same hyperparameters and data as in Section 3.2.[13]

There are two critical differences from our previous experiments: multilingual BERT is trained on much larger amounts of Wikipedia data compared to other LMs used in this work, and the WordPiece vocabulary (Wu et al., 2016) used in the cased multilingual BERT model has been shown to have a distribution skewed toward Latin alphabets (Ács, 2019). These results are thus not directly comparable to those in Figure 1; nevertheless, it is interesting to see that the results obtained with ELMo-like LMs are comparable to and in some cases better than results using a BERT model trained on over a hundred languages. Our results broadly fit with those of Pires et al. (2019), who found that multilingual BERT was useful for zero-shot crosslingual syntactic transfer. In particular, we find nearly no performance benefit from cross-script transfer using BERT in a language pair (English-Japanese) for which they reported poor performance in zero-shot transfer, contrary to our results using Rosita (Section 4.2).

[12] Available at https://github.com/google-research/bert/

[13] AllenNLP version 0.9.0 was used for these experiments.

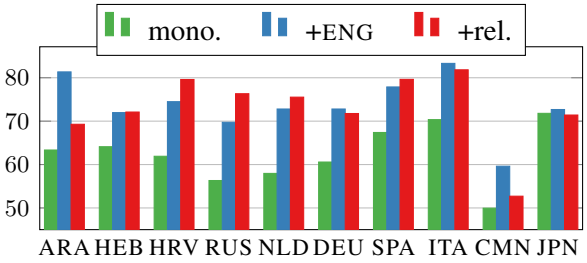


Figure 3: LAS for UD parsing results in a simulated low-resource setting ($(|D_\tau| = 100)$ using multilingual BERT embeddings in place of Rosita. Cf. Figure 1.

5 Decontextual Probe

We saw the success of the joint polyglot training for multilingual CWRs over the retrofitting approach in the previous section. We hypothesize that CWRs from joint training provide useful representations for parsers by inducing *nonlinear* similarity in the vector spaces of different languages that we cannot retrieve with a simple alignment of monolingual pretrained language models. In order to test this hypothesis, we conduct a *decontextual probe* comprised of two steps. The decontextualization step effectively distills CWRs into word type vectors, where each unique word is mapped to exactly one embedding regardless of the context. We then conduct linear transformation-based word translation (Mikolov et al., 2013) on the decontextualized vectors to quantify the degree of crosslingual similarity in the multilingual CWRs.

5.1 Decontextualization

Recall from Section 2 that we produce CWRs from bidirectional LMs with character CNNs and two-layer LSTMs. We propose a method to remove the dependence on context c for the two LSTM layers (the CNN layer is already context-independent by design). During LM training, the hidden states of each layer h_t are computed by the standard LSTM equations:

$$
\begin{aligned}
i_t &= \sigma\left(W_i x_t + + U_i h_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \\
\tilde{c}_t &= \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh\left(c_t\right)
\end{aligned}
$$

Representations	UD	SRL	NER
GloVe	83.78	80.01	83.90
fastText	83.93	80.27	83.40
Decontextualization	86.88	81.41	87.72
ELMo	88.71	82.12	88.65

Table 5: Context independent vs. dependent performance in English. All embeddings are 512-dimensional and trained on the same English corpus of approximately 50M tokens for fair comparisons. We also concatenate 128-dimensional character LSTM representations with the word vectors in every configuration to ensure all models have character input. UD scores are LAS, and SRL and NER are F_1.

We produce contextless vectors from pretrained LMs by removing recursion in the computation (i.e. setting h_{t-1} and c_{t-1} to 0):

$$
\begin{aligned}
i_t &= \sigma\left(W_i x_t + b_i\right) \\
f_t &= \sigma\left(W_f x_t + b_f\right) \\
\tilde{c}_t &= \tanh\left(W_c x_t + b_c\right) \\
o_t &= \sigma\left(W_o x_t + b_o\right) \\
c_t &= i_t \odot \tilde{c}_t \\
h_t &= o_t \odot \tanh\left(c_t\right)
\end{aligned}
$$

This method is fast to compute, as it does not require recurrent computation and only needs to see each word once. This way, each word is associated with a set of exactly three vectors from the three layers.

Performance of decontextualized vectors We perform a brief experiment to find what information is successfully retained by the decontextualized vectors, by using them as inputs to three tasks (in a monolingual English setting, for simplicity). For Universal Dependencies (UD) parsing, semantic role labeling (SRL), and named entity recognition (NER), we used the standard train/development/test splits from UD English EWT (Zeman et al., 2018) and Ontonotes (Pradhan et al., 2013). Following Mulcaire et al. (2019), we use strong existing neural models for each task: Dozat and Manning (2017) for UD parsing, He et al. (2017) for SRL, and Peters et al. (2017) for NER.

Table 5 compares the decontextualized vectors with the original CWRs (ELMo) and the conventional word type vectors, GloVe (Pennington et al., 2014) and fastText (Bojanowski et al., 2017). In all three tasks, the decontextualized vectors substantially improve over fastText and GloVe vectors, and perform nearly on par with contextual ELMo. This suggests that while part of the advantage of CWRs is in the incorporation of context, they also benefit from rich context-*independent* representations present in deeper networks.

Vector	DEU	SPA	FRA	ITA	POR	SWE
fastText	31.6	54.8	56.7	50.2	55.5	43.9
ELMos						
Layer 0	19.7	41.5	41.1	36.9	44.6	27.5
Layer 1	24.4	46.4	47.6	44.2	48.3	36.3
Layer 2	19.9	40.5	41.9	38.1	42.5	30.9
Rosita						
Layer 0	37.9	**56.6**	**58.2**	57.5	**56.6**	50.6
Layer 1	**40.3**	56.3	57.2	**58.1**	56.5	**53.7**
Layer 2	38.8	51.1	52.7	53.6	50.7	50.8

Table 6: Crosslingual alignment results (precision at 1) from decontextual probe. Layers 0, 1, and 2 denote the character CNN, first LSTM, and second LSTM layers in the language models respectively.

5.2 Word Translation Test

Given the decontextualized vectors from each layer of the bidirectional language models, we can measure the crosslingual lexical correspondence in the multilingual CWRs by performing word translation. Concretely, suppose that we have training and evaluation word translation pairs from the source to the target language. Using the same word alignment objective discussed as in Section 2.1, we find a linear transform by aligning the decontextualized vectors for the training source-target word pairs. Then, we apply this linear transform to the decontextualized vector for each source word in the evaluation pairs. The closest target vector is found using the cross-domain similarity local scaling (CSLS) measure (Conneau et al., 2018), which is designed to remedy the hubness problem (where a few "hub" points are nearest neighbors to many other points each) in word translation by normalizing the cosine similarity according to the degree of hubness.

We again take the dictionaries from Conneau et al. (2018) with the given train/test split, and always use English as the target language. For each language, we take all words that appear three times or more in our LM training data and compute decontextualized vectors for them. Word translation is evaluated by choosing the closest vector among the English decontextualized vectors.

5.3 Results

We present word translation results from our decontextual probe in Table 6. We see that the first

LSTM layer generally achieves the best crosslingual alignment both in ELMos and Rosita. This finding mirrors recent studies on layerwise transferability; representations from the first LSTM layer in a language model are most transferable across a range of tasks (Liu et al., 2019). Our decontextual probe demonstrates that the first LSTM layer learns the most generalizable representations not only across tasks but also across languages. In all six languages, Rosita (joint LM training approach) outperforms ELMos (retrofitting approach) and the fastText vectors. This shows that for the polyglot (jointly trained) LMs, there is a preexisting similarity between languages' vector spaces beyond what a linear transform provides. The resulting language-agnostic representations lead to polyglot training's success in low-resource dependency parsing.

6 Further Related Work

In addition to the work mentioned above, much previous work has proposed techniques to transfer knowledge from a high-resource to a low-resource language for dependency parsing. Many of these methods use an essentially (either lexicalized or delexicalized) joint polyglot training setup (e.g., McDonald et al., 2011; Cohen et al., 2011; Duong et al., 2015; Guo et al., 2016; Vilares et al., 2016; Falenska and Çetinoğlu, 2017 as well as many of the CoNLL 2017/2018 shared task participants: Lim and Poibeau (2017); Vania et al. (2017); de Lhoneux et al. (2017); Che et al. (2018); Wan et al. (2018); Smith et al. (2018); Lim et al. (2018)). Some use typological information to facilitate crosslingual transfer (e.g., Naseem et al., 2012; Täckström et al., 2013; Zhang and Barzilay, 2015; Wang and Eisner, 2016; Rasooli and Collins, 2017; Ammar et al., 2016). Others use bitext (Zeman et al., 2018), manually-specified rules (Naseem et al., 2012), or surface statistics from gold universal part of speech (Wang and Eisner, 2018a,b) to map the source to target. The methods examined in this work to produce multilingual CWRs do not rely on such external information about the languages, and instead use relatively abundant LM data to learn crosslinguality that abstracts away from typological divergence.

Recent work has developed several probing methods for (monolingual) contextual representations (Liu et al., 2019; Hewitt and Manning, 2019; Tenney et al., 2019). Wada and Iwata (2018) showed that the (contextless) input and output word vectors in a polyglot word-based language model manifest a certain level of lexical correspondence between languages. Our decontextual probe demonstrated that the *internal* layers of polyglot language models capture crosslinguality and produce useful multilingual CWRs for downstream low-resource dependency parsing.

7 Conclusion

We assessed recent approaches to multilingual contextual word representations, and compared them in the context of low-resource dependency parsing. Our parsing results illustrate that a joint training approach for polyglot language models outperforms a retrofitting approach of aligning monolingual language models. Our decontextual probe showed that jointly trained LMs learn a better crosslingual lexical correspondence than the one produced by aligning monolingual language models or word type vectors. Our results provide a strong basis for multilingual representation learning and for further study of crosslingual transfer in a low-resource setting beyond dependency parsing.

Acknowledgments

The authors thank Nikolaos Pappas and Tal Schuster as well as the anonymous reviewers for their helpful feedback. This research was funded in part by NSF grant IIS-1562364, a Google research award to NAS, the Funai Overseas Scholarship to JK, and the NVIDIA Corporation through the donation of a GeForce GPU.

References

Judit Ács. 2019. Exploring BERT's vocabulary.

Wasi Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. In *Proc. of NAACL-HLT*.

Hanan Aldarmaki and Mona Diab. 2019. Context-aware cross-lingual mapping. In *Proc. of NAACL-HLT*.

Waleed Ammar. 2016. *Towards a Universal Analyzer of Natural Languages*. Ph.D. thesis, Carnegie Mellon University.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *TACL*, 4.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. arXiv:1312.3005.

Shay B Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proc. of EMNLP*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proc. of ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT*.

Timothy Dozat and Christopher Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proc. of ICLR*.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the conll 2017 shared task. In *Proc. of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proc. of ACL-IJCNLP*.

Agnieszka Falenska and Özlem Çetinoğlu. 2017. Lexicalized vs. delexicalized parsing in low-resource scenarios. In *Proc. of IWPT*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proc. of NAACL-HLT*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proc. of NLP-OSS*.

Daniela Gerz, Ivan Vulić, Edoardo Maria Ponti, Roi Reichart, and Anna Korhonen. 2018. On the relation between linguistic typology and (limitations of) multilingual language modeling. In *Proc. of EMNLP*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proc. of ACL-IJCNLP*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proc. of AAAI*.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proc. of ACL*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proc. of NAACL-HLT*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. ADAM: A Method for Stochastic Optimization. In *Proc. of ICLR*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. In *Proc. of NeurIPS*.

Miryam de Lhoneux, Johannes Bjerva, Isabelle Augenstein, and Anders Søgaard. 2018. Parameter sharing between dependency parsers for related languages. In *Proc. of EMNLP*.

Miryam de Lhoneux, Yan Shao, Ali Basirat, Eliyahu Kiperwasser, Sara Stymne, Yoav Goldberg, and Joakim Nivre. 2017. From raw text to universal dependencies - look, no tags! In *Proc. of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

KyungTae Lim, Cheoneum Park, Changki Lee, and Thierry Poibeau. 2018. SEx BiST: A multi-source trainable parser with deep contextualized lexical representations. In *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

KyungTae Lim and Thierry Poibeau. 2017. A system for multilingual dependency parsing based on bidirectional LSTM feature representations. In *Proc. of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proc. of NAACL-HLT*.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proc. of ACL*.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proc. of ACL*.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proc. of EMNLP*.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. arXiv:1309.4168.

Phoebe Mulcaire, Jungo Kasai, and Noah A. Smith. 2019. Polyglot contextual representations improve crosslingual transfer. In *Proc. of NAACL-HLT*.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proc. of ACL*.

Joakim Nivre, Mitchell Abrams, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Gashaw Arutie, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Sandra Bellato, Kepa Bengoetxea, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Rogier Blokland, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Yongseok Cho, Jayeol Chun, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Carly Dickerson, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Tomaž Erjavec, Aline Etienne, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Na-Rae Han, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Jena Hwang, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Sookyoung Kwak, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phng Lê H`ông, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, KyungTae Lim, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lng Nguy˜ên Thị, Huy`ên Nguy˜ên Thị Minh, Vitaly Nikolaev, Rattima Nitisaroj, Hanna Nurmi, Stina Ojala, Adédayọ̀ Olúòkun, Mai Omura, Petya Osenova, Robert Östling, Lilja Øvrelid, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Siyao Peng, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Thierry Poibeau, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Andriela Rääbis, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Carlos Ramisch, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Michael Rießler, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Valentin Roca, Olga Rudina, Shoval Sadde, Shadi Saleh, Tanja Samardžić, Stephanie Samson, Manuela Sanguinetti, Baiba Saulīte, Yanin Sawanakunanon, Nathan Schneider, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Muh Shohibussirri, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Isabela Soares-Bastos, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Yuta Takahashi, Takaaki Tanaka, Isabelle Tellier, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Veronika Vincze, Lars Wallin, Jonathan North Washington, Seyi Williams, Mats Wirén, Tsegay Woldemariam, Tak-sum Wong, Chunxiao Yan, Marat M. Yavrumyan, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, Manying Zhang, and Hanzhi Zhu. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Robert Östling and Jörg Tiedemann. 2017. Continuous multilinguality with language vectors. In *Proc. of EACL*.

Jeffrey Pennington, Richard Socher, and Christo-

pher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proc. of ACL*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL-HLT*.

Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual BERT? In *Proc. of ACL*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proc. of CoNLL*.

Anton Ragni, Edgar Dakin, Xie Chen, Mark J. F. Gales, and Kate Knill. 2016. Multi-language neural network language models. In *Proc. of INTERSPEECH*.

Mohammad Sadegh Rasooli and Michael Collins. 2017. Cross-lingual syntactic transfer with limited resources. *TACL*, 5.

Rudolf Rosa and David Mareček. 2018. CUNI x-ling: Parsing under-resourced languages in CoNLL 2018 UD shared task. In *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Tal Schuster, Ori Ram, Regina Barzilay, and Amir Globerson. 2019. Cross-lingual alignment of contextual word embeddings, with applications to zero-shot dependency parsing. In *Proc. of NAACL-HLT*.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 treebanks, 34 models: Universal dependency parsing with multi-treebank models. In *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proc. of NAACL-HLT*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proc. of ACL*.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *Proc. of NAACL-HLT*.

Clara Vania, Xingxing Zhang, and Adam Lopez. 2017. UParse: the Edinburgh system for the CoNLL 2017 UD shared task. In *Proc. of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

David Vilares, Carlos Gómez-Rodríguez, and Miguel A. Alonso. 2016. One model, two languages: training bilingual parsers with harmonized treebanks. In *Proc. of ACL*.

Takashi Wada and Tomoharu Iwata. 2018. Unsupervised cross-lingual word embedding by multilingual neural language models. arXiv:1809.02306.

Hui Wan, Tahira Naseem, Young-Suk Lee, Vittorio Castelli, and Miguel Ballesteros. 2018. IBM research at the CoNLL 2018 shared task on multilingual parsing. In *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Dingquan Wang and Jason Eisner. 2016. The galactic dependencies treebanks: Getting more data by synthesizing new languages. *TACL*, 4.

Dingquan Wang and Jason Eisner. 2018a. Surface statistics of an unknown language indicate how to parse it. *TACL*, 6.

Dingquan Wang and Jason Eisner. 2018b. Synthetic data made to order: The case of parsing. In *Proc. of EMNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. arxiv:1212.5701.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies. In *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proc. of EMNLP*.

Improving Pre-Trained Multilingual Models with Vocabulary Expansion

Hai Wang[1*] **Dian Yu**[2] **Kai Sun**[3*] **Jianshu Chen**[2] **Dong Yu**[2]
[1]Toyota Technological Institute at Chicago, Chicago, IL, USA
[2]Tencent AI Lab, Bellevue, WA, USA [3]Cornell, Ithaca, NY, USA
haiwang@ttic.edu, ks985@cornell.edu,
{yudian,jianshuchen,dyu}@tencent.com

Abstract

Recently, pre-trained language models have achieved remarkable success in a broad range of natural language processing tasks. However, in multilingual setting, it is extremely resource-consuming to pre-train a deep language model over large-scale corpora for each language. Instead of exhaustively pre-training monolingual language models independently, an alternative solution is to pre-train a powerful multilingual deep language model over large-scale corpora in hundreds of languages. However, the vocabulary size for each language in such a model is relatively small, especially for low-resource languages. This limitation inevitably hinders the performance of these multilingual models on tasks such as sequence labeling, wherein in-depth token-level or sentence-level understanding is essential.

In this paper, inspired by previous methods designed for monolingual settings, we investigate two approaches (i.e., joint mapping and mixture mapping) based on a pre-trained multilingual model BERT for addressing the out-of-vocabulary (OOV) problem on a variety of tasks, including part-of-speech tagging, named entity recognition, machine translation quality estimation, and machine reading comprehension. Experimental results show that using mixture mapping is more promising. To the best of our knowledge, this is the first work that attempts to address and discuss the OOV issue in multilingual settings.

1 Introduction

It has been shown that performance on many natural language processing tasks drops dramatically on held-out data when a significant percentage of words do not appear in the training data, i.e., out-of-vocabulary (OOV) words (Søgaard and Johannsen, 2012; Madhyastha et al., 2016). A higher OOV rate (i.e., the percentage of the unseen words in the held-out data) may lead to a more severe performance drop (Kaljahi et al., 2015). OOV problems have been addressed in previous works under monolingual settings, through replacing OOV words with their semantically similar in-vocabulary words (Madhyastha et al., 2016; Kolachina et al., 2017) or using character/word information (Kim et al., 2016, 2018; Chen et al., 2018) or subword information like byte pair encoding (BPE) (Sennrich et al., 2016; Stratos, 2017).

Recently, fine-tuning a pre-trained deep language model, such as Generative Pre-Training (GPT) (Radford et al., 2018) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018), has achieved remarkable success on various downstream natural language processing tasks. Instead of pre-training many monolingual models like the existing English GPT, English BERT, and Chinese BERT, a more natural choice is to develop a powerful multilingual model such as the multilingual BERT.

However, all those pre-trained models rely on language modeling, where a common trick is to tie the weights of softmax and word embeddings (Press and Wolf, 2017). Due to the expensive computation of softmax (Yang et al., 2017) and data imbalance across different languages, the vocabulary size for each language in a multilingual model is relatively small compared to the monolingual BERT/GPT models, especially for low-resource languages. Even for a high-resource language like Chinese, its vocabulary size 10k in the multilingual BERT is only half the size of that in the Chinese BERT. Just as in monolingual settings, the OOV problem also hinders the performance of a multilingual model on tasks that are sensitive to token-level or sentence-level information. For ex-

* This work was done when H. W. and K. S. were at Tencent AI Lab, Bellevue, WA.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 316–327
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

ample, in the POS tagging problem (Table 2), 11 out of 16 languages have significant OOV issues (OOV rate $\geq 5\%$) when using multilingual BERT.

According to previous work (Radford et al., 2018; Devlin et al., 2018), it is time-consuming and resource-intensive to pre-train a deep language model over large-scale corpora. To address the OOV problems, instead of pre-training a deep model with a large vocabulary, we aim at enlarging the vocabulary size when we fine-tune a pre-trained multilingual model on downstream tasks.

We summarize our contributions as follows: (i) We investigate and compare two methods to alleviate the OOV issue. To the best of our knowledge, this is the first attempt to address the OOV problem in multilingual settings. (ii) By using English as an interlingua, we show that bilingual information helps alleviate the OOV issue, especially for low-resource languages. (iii) We conduct extensive experiments on a variety of token-level and sentence-level downstream tasks to examine the strengths and weaknesses of these methods, which may provide key insights into future directions[1].

2 Approach

We use the multilingual BERT as the pre-trained model. We first introduce the pre-training procedure of this model (Section 2.1) and then introduce two methods we investigate to alleviate the OOV issue by expanding the vocabulary (Section 2.2). Note that these approaches are not restricted to BERT but also applicable to other similar models.

2.1 Pre-Trained BERT

Compared to GPT (Radford et al., 2018) and ELMo (Peters et al., 2018), BERT (Devlin et al., 2018) uses a **bidirectional** transformer, whereas GPT pre-trains a left-to-right transformer (Liu et al., 2018); ELMo (Peters et al., 2018) independently trains left-to-right and right-to-left LSTMs (Peters et al., 2017) to generate representations as additional features for end tasks.

In the pre-training stage, Devlin et al. (2018) use two objectives: masked language model (LM) and next sentence prediction (NSP). In masked LM, they randomly mask some input tokens and then predict these masked tokens. Compared to unidirectional LM, masked LM enables representations to fuse the context from both directions. In the NSP task, given a certain sentence, it aims to predict the next sentence. The purpose of adding the NSP objective is that many downstream tasks such as question answering and language inference require sentence-level understanding, which is not directly captured by LM objectives.

After pre-training on large-scale corpora like Wikipedia and BookCorpus (Zhu et al., 2015), we follow recent work (Radford et al., 2018; Devlin et al., 2018) to fine-tune the pre-trained model on different downstream tasks with minimal architecture adaptation. We show how to adapt BERT to different downstream tasks in Figure 1 (left).

2.2 Vocabulary Expansion

Devlin et al. (2018) pre-train the multilingual BERT on Wikipedia in 102 languages, with a shared vocabulary that contains 110k subwords calculated from the WordPiece model (Wu et al., 2016). If we ignore the shared subwords between languages, on average, each language has a 1.1k vocabulary, which is significantly smaller than that of a monolingual pre-trained model such as GPT (40k). The OOV problem tends to be less serious for languages (e.g., French and Spanish) that belong to the same language family of English. However, this is not always true, especially for morphologically rich languages such as German (Ataman and Federico, 2018; Lample et al., 2018). OOV problem is much more severe in low-resource scenarios, especially when a language (e.g., Japanese and Urdu) uses an entirely different character set from high-resource languages.

We focus on addressing the OOV issue at subword level in multilingual settings. Formally, suppose we have an embedding E_{bert} extracted from the (non-contextualized) embedding layer in the multilingual BERT (i.e., the first layer of BERT). And suppose we have another set of (non-contextualized) sub-word embeddings $\{E_{l_1}, E_{l_2}, \ldots, E_{l_n}\} \cup \{E_{en}\}$, which are pre-trained on large corpora using any standard word embedding toolkit. Specifically, E_{en} represents the pre-trained embedding for English, and E_{l_i} represents the pre-trained embedding for non-English language l_i at the subword level. We denote the vocabulary of E_{bert}, E_{en}, and E_{l_i} by V_{bert}, V_{en}, and V_{l_i}, respectively. For each subword w in V_{bert}, we use $E_{bert}(w)$ to denote the pre-trained embedding of word w in E_{bert}. $E_{l_i}(\cdot)$ and $E_{en}(\cdot)$ are defined in a similar way as $E_{bert}(\cdot)$. For

[1]Improved models will be available at https://github.com/sohuren/multilingul-bert.

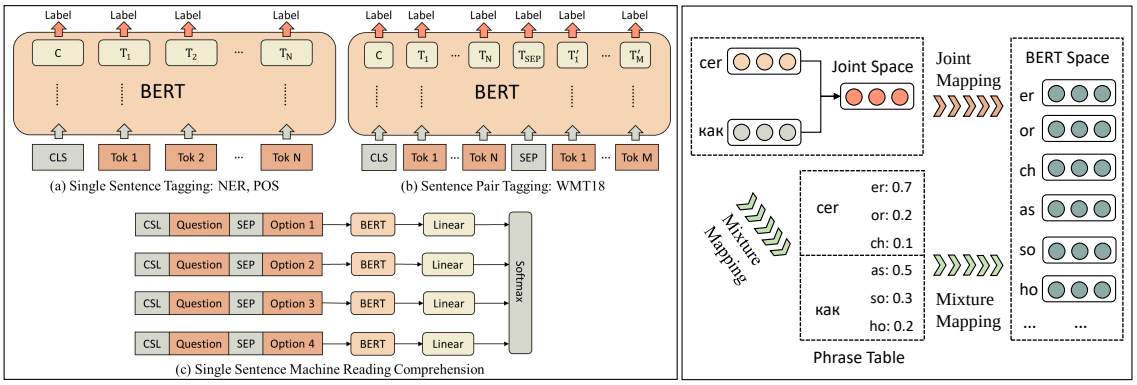


Figure 1: Left: fine-tuning BERT on different kinds of end tasks. Right: illustration of joint and mixture mapping (in this example, during mixture mapping, we represent $\boldsymbol{e}(cer) = 0.7 * \boldsymbol{e}(er) + 0.2 * \boldsymbol{e}(or) + 0.1 * \boldsymbol{e}(ch)$).

each non-English language $l \in \{l_1, l_2, \ldots, l_n\}$, we aim to enrich E_{bert} with more subwords from the vocabulary in E_{l_i} since E_{l_i} contains a larger vocabulary of language l_i compared to E_{bert}.

As there is no previous work to address multilingual OOV issues, inspired by previous solutions designed for monolingual settings, we investigate the following two methods, and all of them can be applied at both word/subword level, though we find subword-level works better (Section 3).

Joint Mapping For each non-English language l, we first construct a joint embedding space E'_l through mapping E_l to E_{en} by an orthogonal mapping matrix B_l (i.e., $E'_l = E_l B_l$). When a bilingual dictionary $f_l : V_l \rightarrow V_{en}$ is available or can be constructed based on the shared common subwords (Section 3.1), we obtain B_l by minimizing:

$$\sum_{w' \in V_l \cap \{w : f_l(w) \in V_{en}\}} \left\| E_l(w') B_l - E_{en}(f_l(w')) \right\|_F^2$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Otherwise, for language pair (e.g., English-Urdu) that meets neither of the above two conditions, we obtain B_l by an unsupervised word alignment method from MUSE (Conneau et al., 2018).

We then map E'_l to E_{bert} by an orthogonal mapping matrix A'_l, which is obtained by minimizing

$$\sum_{w \in f_l(V_l) \cap V_{bert}} \left\| E'_l(w) A'_l - E_{bert}(w) \right\|_F^2$$

We denote this method by M_J in our discussion below, where the subscript J stands for "joint".

Mixture Mapping Following the work of Gu et al. (2018) where they use English as *"universal tokens"* and map all other languages to English to obtain the subword embeddings, we represent each subword in E'_l (described in joint mapping) as a mixture of English subwords where those English subwords are already in the BERT vocab V_{bert}. This method, denoted by M_M, is also a joint mapping without the need for learning the mapping from E'_l to E_{bert}. Specifically, for each $w \in V_l$, we obtain its embedding $\boldsymbol{e}(w)$ in the BERT embedding space E_{bert} as follows.

$$\boldsymbol{e}(w) = \sum_{u \in \mathcal{T}(w)} p(u \mid w) E_{bert}(u)$$

where $\mathcal{T}(w)$ is a set to be defined later, and the mixture coefficient $p(u|w)$ is defined by

$$p(u \mid w) = \frac{\exp(\text{CSLS}(E_l(u), E_{en}(w)))}{\sum_{v \in \mathcal{T}(w)} \exp(\text{CSLS}(E_l(v), E_{en}(w)))}$$

where CSLS refers to the distance metric Cross-domain Similarity Local Scaling (Conneau et al., 2018). We select five $v \in V_{en} \cup V_{bert}$ with the highest $\text{CSLS}(E_l(v), E_{en}(w))$ to form set $\mathcal{T}(w)$. In all our experiments, we set the number of nearest neighbors in CSLS to 10. We refer readers to Conneau et al. (2018) for details. Figure 1 (right) illustrates the joint and mixture mapping.

3 Experiment

3.1 Experiment Settings

We obtain the pre-trained embeddings of a specific language by training fastText (Bojanowski et al., 2017) on Wikipedia articles in that language, with context window 5 and negative sampling 5. Before training, we first apply BPE (Sennrich et al., 2016) to tokenize the corpus with subword vocabulary size 50k. For joint mapping method M_J,

we use bilingual dictionaries provided by Conneau et al. (2018). For a language pair where a bilingual dictionary is not easily available, if two languages share a significant number of common subwords (this often happens when two languages belong to the same language family), we construct a bilingual dictionary based on the assumption that identical subwords have the same meaning (Søgaard et al., 2018). We add all unseen subwords from 50k vocabulary to BERT. We define a word as an OOV word once it cannot be represented as a single word. For example, in BERT, the sentence *"Je sens qu' entre ça et les films de médecins et scientifiques"* is represented as *"je sens qu ##' entre [UNK] et les films de [UNK] et scientifiques"*, where **qu'** is an OOV word since it can only be represented by two subword units: **qu** and **##'**, but it is not OOV at subword level; **ça** and **médecins** cannot be represented by any single word or combination of subword units, and thus they are OOV at both word and subword level.

We use the **multilingual BERT** with default parameters in all our experiments, except that we tune the batch size and training epochs. To have a thorough examination about the pros and cons of the explored methods, we conduct experiments on a variety of token-level and sentence-level classification tasks: part of speech (POS) tagging, named entity recognition (NER), machine translation quality estimation, and machine reading comprehension. See more details in each subsection.

3.2 Discussions about Mapping Methods

Previous work typically assumes a linear mapping exists between embedding spaces of different languages if their embeddings are trained using similar techniques (Xing et al., 2015; Madhyastha et al., 2016). However, it is difficult to map embeddings learned with different methods (Søgaard et al., 2018). Considering the differences between BERT and fastText: e.g., the objectives, the way to differentiate between different subwords, and the much deeper architecture of BERT, it is very unlikely that the (non-contextualized) BERT embedding and fastText embedding reside in the same geometric space. Besides, due to that BERT has a relatively smaller vocabulary for each language, when we map a pre-trained vector to its portion in BERT indirectly as previous methods, the supervision signal is relatively weak, especially for low-resource languages. In our experiment, we find that the accuracy of the mapping from our pre-trained English embedding to multilingual BERT embedding (English portion) is lower than 30%. In contrast, the accuracy of the mapping between two regular English embeddings that are pre-trained using similar methods (e.g., CBOW or SkipGram (Mikolov et al., 2013)) could be above 95% (Conneau et al., 2018).

Besides the **joint mapping** method M_J (Section 2.2), another possible method that could be used for OOV problem in multilingual setting is that, for each language l, we map its pre-trained embedding space E_l to embedding E_{bert} by an orthogonal mapping matrix A_l, which is obtained by minimizing $\sum_{w \in V_l \cap V_{bert}} \|E_l(w)A_l - E_{bert}(w)\|_F^2$. This approach is similar to (Madhyastha et al., 2016), and is referred as **independent mapping** method below. However, we use examples to demonstrate why these kind of methods are less promising. In Table 1, the first two rows are results obtained by mapping our pre-trained English embedding (using fastText) to the (non-contextualized) BERT embedding. In this new unified space, we align words with CSLS metric, and for each subword that appears in English Wikipedia, we seek its closest neighbor in the BERT vocabulary. Ideally, each word should find itself if it exists in the BERT vocabulary. However, this is not always true. For example, although *"however"* exists in the BERT vocabulary, independent mapping fails to find it as its own closest neighbor. Instead, it incorrectly maps it to irrelevant Chinese words "盘" (*"plate"*) and "龙" (*"dragon"*). A similar phenomenon is observed for Chinese. For example, "册" is incorrectly aligned to Chinese words "书" and "卷".

Source Lang	Source	Target	probability
English	however	盘(plate)	0.91
	however	龙(dragon)	0.05
Chinese	册(booklet)	书(book)	0.49
	册(booklet)	卷(volume)	0.46

Table 1: Alignment from Independent Mapping.

Furthermore, our POS tagging experiments (Section 3.3) further show that joint mapping M_J does not improve (or even hurt) the performance of the multilingual BERT. Therefore, we use **mixture mapping** M_M to address and discuss OOV issues in the remaining sections.

	BTS♣	BiLSTM♢	FREQ♢	BERT	BERT_{oov}	$\text{BERT}_{\text{oovR}}$	$\text{BERT}_{\text{oovMJ}}$	OOV_{w}	OOV_{sw}
ar	-	98.23	90.06	53.34	**56.70**	56.57	56.23	89.8	70.6
bg	97.84	98.23	90.06	**98.70**	98.22	94.41	97.21	45.7	1.2
da	95.52	96.16	96.35	**97.16**	96.53	94.15	94.85	38.9	2.8
de	92.87	93.51	93.38	93.58	**93.81**	91.77	93.12	43.2	5.6
es	95.80	95.67	95.74	96.04	**96.92**	95.10	95.77	29.4	6.0
fa	96.82	97.60	97.49	95.62	94.90	94.35	**95.82**	35.6	6.5
fi	95.48	95.74	95.85	87.72	**93.35**	84.75	89.39	64.9	10.4
fr	95.75	96.20	96.11	95.17	**96.59**	94.84	95.24	33.9	10.3
hr	-	96.27	96.82	95.03	**96.49**	89.87	93.48	49.5	8.3
it	97.56	97.90	97.95	**98.22**	98.00	97.63	97.85	30.3	2.3
nl	-	92.82	93.30	**93.89**	92.89	91.30	91.71	35.5	0.3
no	-	98.06	98.03	**97.25**	95.98	94.21	95.83	38.7	4.4
pl	-	97.63	97.62	91.62	**95.95**	87.50	92.56	56.5	13.6
pt	-	97.94	97.90	96.66	**97.63**	95.93	96.90	34.0	8.3
sl	-	96.97	96.84	95.02	**96.91**	89.55	93.97	49.2	7.8
sv	95.57	96.60	96.69	91.23	**96.66**	90.45	91.92	48.2	17.7
average	-	96.60	95.64	92.27	93.60	90.15	92.20	45.2	11.0

Table 2: POS tagging accuracy (%) on the Universal Dependencies v1.2 dataset. BERT_{oov}: BERT with method M_M. $\text{BERT}_{\text{oovR}}$: BERT with randomly picked embedding from BERT. $\text{BERT}_{\text{oovMJ}}$: BERT with method M_J. OOV_{w}: word-level OOV rate. OOV_{sw}: subword-level OOV rate. ♣: Gillick et al. (2016), ♢: Plank et al. (2016).

Approach	**Precision**	**Recall**	**F1 score**
DomainMask (Peng and Dredze, 2017a)	60.8	44.9	51.7
Linear Projection (Peng and Dredze, 2017a)	67.2	41.2	51.1
Updates (Peng and Dredze, 2017b)	-	-	56.1
Updates (Peng and Dredze, 2017b)	-	-	59.0
BERT	56.6	61.7	59.0
BERT_{oov}	60.2	62.8	**61.4**
BERT_{zh}	63.4	70.8	**66.9**

Table 3: Performance of various models on the test set of Weibo NER. BERT_{zh}: Chinese BERT pre-trained over Chinese Wikipedia. We use scripts *conlleval* for evaluation on NER.

3.3 Monolingual Sequence Labeling Tasks

POS Tagging: We use the Universal Dependencies v1.2 data (McDonald et al., 2013). For languages with token segmentation ambiguity, we use the gold segmentation following Plank et al. (2016). We consider languages that have sufficient training data and filter out languages that have unsatisfying embedding alignments with English (accuracy is lower than 20.0% measured by word alignment accuracy or 0.25 by unsupervised metric in MUSE (Conneau et al., 2018)). Finally, we keep 16 languages. We use the original multilingual BERT (without using CRF (Lafferty et al., 2001) on top of it for sequence labeling) to tune hyperparameters on the dev set and use the fixed hyperparameters for the expanded multilingual model. We do not tune the parameters for each model separately. As shown in Table 2, at both the word and subword level, the OOV rate in this dataset is quite high. Mixture mapping improves the accuracy on 10 out of 16 languages, leading to a 1.97% absolute gain in average. We discuss the influence of alignments in Section 3.6.

Chinese NER: We are also interested in investigating the performance gap between the expanded multilingual model and a monolingual BERT that is pre-trained on a large-scale monolingual corpus. Currently, pre-trained monolingual BERT models are available in English and Chinese. As English has been used as the interlingua, we compare the expanded multilingual BERT and the Chinese BERT on a Chinese NER task, evaluated on the Weibo NER dataset constructed from social media by Peng and Dredze (2015). In the training set, the token-level OOV rate is 2.17%, and the subword-level OOV rate is 0.54%. We tune the hyperparameters of each model based on the development set separately and then use the best hyperparameters of each model for evaluation on the test set.

As shown in Table 3, the expanded model outperforms the multilingual BERT on the Weibo NER dataset. We boost the F1 score from 59.0%

to 61.4%. Compared to the Chinese BERT (66.9%), there still exists a noticeable performance gap. One possible reason could be the grammatical differences between Chinese and English. As BERT uses the language model loss function for pre-training, the pre-trained Chinese BERT could better capture the language-specific information comapred to the multilingual BERT.

3.4 Code-Mixed Sequence Labeling Tasks

As the multilingual BERT is pre-trained over 102 languages, it should be able to handle code-mixed texts. Here we examine its performance and the effectiveness of the expanded model in mixed language scenarios, using two tasks as case studies.

Code-Switch Challenge: We first evaluate on the CALCS-2018 code-switched task (Aguilar et al., 2018), which contains two NER tracks on Twitter social data: mixed English&Spanish (en-es) and mixed Modern Standard Arabic&Egyptian (ar-eg). Compared to traditional NER datasets constructed from news, the dataset contains a significant portion of uncommon tokens like hashtags and abbreviations, making it quite challenging. For example, in the en-es track, the token-level OOV rate is 44.6%, and the subword-level OOV rate is 3.1%; in the ar-eg track, the token-level OOV rate is 64.0%, and the subword-level OOV rate is 6.0%. As shown in Table 4, on ar-eg, we boost the F1 score from 74.7% to 77.3%. However, we do not see similar gains on the en-es dataset, probably because that English and Spanish share a large number of subwords, and adding too many new subwords might prevent the model from utilizing the well pre-trained subwords embedding. See Section 3.6 for more discussions.

	en-es			ar-eg		
Model	Prec	Rec	F1	Prec	Rec	F1
FAIR♣	-	-	62.4	-	-	71.6
IIT♣	-	-	63.8	-	-	-
FAIR♢	-	-	67.7	-	-	**81.4**
BERT	72.7	**63.6**	**67.8**	73.8	75.6	74.7
BERT_{oov}	**74.2**	60.9	66.9	**76.9**	**77.8**	**77.3**

Table 4: Accuracy (%) on the code-switch challenge. The top two rows are based on the test set, and the bottom three rows are based on the development set. ♣: results from Aguilar et al. (2018). ♢: results from Wang et al. (2018).

Machine Translation Quality Estimation: All previous experiments are based on well-curated data. Here we evaluate the expanded model on a language generation task, where sometimes the generated sentences are out-of-control.

We choose the automatic Machine Translation Quality Estimation task and use Task 2 – word-level quality estimation – in WMT18 (Bojar et al., 2018). Given a source sentence and its translation (i.e., target), this task aims to estimate the translation quality (*"BAD"* or *"OK"*) at each position: e.g., each token in the source and target sentence, each **gap** in the target sentence. We use English to German (en-de) SMT translation. On all three categories, the expanded model consistently outperforms the original multilingual BERT (Table 5)[2].

3.5 Sequence Classification Tasks

Finally, we evaluate the expanded model on sequence classification in a mixed-code setting, where results are less sensitive to unseen words.

Code-Mixed Machine Reading Comprehension: We consider the mixed-language machine reading comprehension task. Since there is no such public available dataset, we construct a new Chinese-English code-mixed machine reading comprehension dataset based on 37,436 unduplicated utterances obtained from the transcriptions of a Chinese and English mixed speech recognition corpus King-ASR-065-1[3]. We generate a multiple-choice machine reading comprehension problem (i.e., a question and four answer options) for each utterance. A question is an utterance with an English text span removed (we randomly pick one if there are multiple English spans) and the correct answer option is the removed English span. Distractors (i.e., wrong answer options) come from the top three closest English text spans, which appear in the corpus, based on the cosine similarity of word embeddings trained on the same corpus. For example, given a question "突然听到 21___，那强劲的鼓点，那一张张脸。" (*"Suddenly I heard 21___, and the powerful drum beats reminded me of the players."*) and four answer options { *"forever"*, *"guns"*, *"jay"*, *"twins"* }, the task is to select the correct answer option *"guns"* (*"21 Guns"* is a song by the American rock band *Green Day*). We split the dataset into training, development, and testing of size 36,636, 400, 400, respectively. An-

[2]Our evaluation is based on the development set since the test set is only available to participants, and we could not find the submission teams' performance on the developmenet set.

[3]http://kingline.speechocean.com.

Model	Words in MT			Gaps in MT			Words in SRC		
	F1-BAD	F1-OK	F1-multi	F1-BAD	F1-OK	F1-multi	F1-BAD	F1-OK	F1-multi
Fan et al. (2018)	0.68	0.92	**0.62**	-	-	-	-	-	-
Fan et al. (2018)	0.66	0.92	0.61	0.51	0.98	**0.50**	-	-	-
SHEF-PT♣	0.51	0.85	0.43	0.29	0.96	0.28	0.42	0.80	0.34
BERT	0.58	0.91	0.53	0.47	0.98	0.46	0.48	0.90	0.43
BERT_{oov}	0.60	0.91	**0.55**	0.50	0.98	**0.49**	0.49	0.90	**0.44**

Table 5: WMT18 Quality Estimation Task 2 for the en→de SMT dataset. ♣: result from Specia et al. (2018). **MT**: machine translation, e.g., target sentence, **SRC**: source sentence. F1-OK: F1 score for *"OK"* class; F1-BAD: F1 score for *"BAD"* class; F1-multi: multiplication of F1-OK and F1-BAD.

notators manually clean and improve the quality problems by generating more confusing distractors in the dev and testing sets to guarantee that these problems are error-free and challenging.

In this experiment, for each BERT model, we follow its default hyperparameters. As shown in Table 6, the expanded model improves the multilingual BERT (38.1%) by 1.2% in accuracy. Human performance (81.4%) indicates that this is not an easy task even for human readers.

Model	Accuracy	
	Development	Test
BERT_{en}	38.2	37.3
BERT	38.7	38.1
BERT_{oov}	39.4	**39.3**
BERT_{zh}	40.0	**45.0**

Table 6: Accuracy (%) of models on the code-mixed reading comprehension dataset. BERT_{en}: pre-trained English BERT. BERT_{zh}: pre-trained Chinese BERT.

3.6 Discussions

In this section, we first briefly investigate whether the performance boost indeed comes from the reduction of OOV and then discuss the strengths and weaknesses of the methods we investigate.

First, we argue that it is essential to alleviate the OOV issue in multilingual settings. Taking the POS tagging task as an example, we find that most errors occur at the OOV positions (Table 7 in Section 3.3). In the original BERT, the accuracy of OOV words is much lower than that of non-OOV words, and we significantly boost the accuracy of OOV words with the expanded BERT. All these results indicate that the overall improvement mostly comes from the reduction of OOV.

We also observe that the following factors may influence the performance of the expanded model. **Subwords**: When expanding the vocabulary, it is critical to add only frequent subwords. Currently, we add all unseen subwords from the 50k vocabulary (Section 3.1), which may be not an optimal choice. Adding too many subwords may prevent the model from utilizing the information from pre-trained subword embedding in BERT, especially when there is a low word-level overlap between the training and test set.

Lang	BERT		BERT_{oov}	
	non-OOV	OOV	non-OOV	OOV
fi	98.1	81.3	98.5	90.2
fr	97.0	90.2	97.2	95.6
hr	97.8	91.9	97.7	94.5
pl	98.8	84.6	99.0	93.2
pt	98.8	91.5	98.6	94.8
sl	98.6	91.6	98.7	95.1
sv	97.4	82.9	98.2	94.8
average	98.1	87.7	98.3	94.0

Table 7: POS tagging accuracy (%) for OOV tokens and non-OOV tokens on the Universal Dependencies v1.2 dataset, where the OOV/non-OOV are defined at word level with the original BERT vocabulary.

Language: We also find that languages can influence the performance of the vocabulary expansion through the following two aspects: the alignment accuracy and the closeness between a language and English. For languages that are closely related to English such as French and Dutch, it is relatively easy to align their embeddings to English as most subword units are shared (Søgaard et al., 2018; Conneau et al., 2018). In such case, the BERT embedding already contains sufficient information, and therefore adding additional subwords may hurt the performance. On the other hand, for a distant language such as Polish (Slavic family), which shares some subwords with English (Germanic family), adding subwords to BERT brings performance improvements. In the meantime, as Slavic and Germanic are two subdivisions of the Indo-European languages, we find that the embedding alignment methods per-

form reasonably well. For these languages, vocabulary expansion is usually more effective, indicated by POS tagging accuracies for Polish, Portuguese, and Slovenian (Table 2). For more distant languages like Arabic (Semitic family) that use different character sets, it is necessary to add additional subwords. However, as the grammar of such a language is very different from that of English, how to accurately align their embeddings is the main bottleneck.

Task: We see more significant performance gains on NER, POS and MT Quality Estimation, possibly because token-level understanding is more critical for these tasks, therefore alleviating OOV helps more. In comparison, for sequence level classification tasks such as machine reading comprehension (Section 3.5), OOV issue is less severe since the result is based on the entire sentence.

4 Related Work

OOV poses challenges for many tasks (Pinter et al., 2017) such as machine translation (Razmara et al., 2013; Sennrich et al., 2016) and sentiment analysis (Kaewpitakkun et al., 2014). Even for tasks such as machine reading comprehension that are less sensitive to the meanings of each word, OOV still hurts the performance (Chu et al., 2017; Zhang et al., 2018). We now discuss previous methods in two settings.

4.1 Monolingual Setting

Most previous work address the OOV problems in monolingual settings. Before more fine-grained encoding schema such as BPE (Sennrich et al., 2016) is proposed, prior work mainly focused on OOV for token-level representations (Taylor et al., 2011; Kolachina et al., 2017). Besides simply assigning random embeddings to unseen words (Dhingra et al., 2017) or using an unique symbol to replace all these words with a shared embedding (Hermann et al., 2015), a thread of research focuses on refining the OOV representations based on word-level information, such as using similar in-vocabulary words (Luong et al., 2015; Cho et al., 2015; Tafforeau et al., 2015; Li et al., 2016), mapping initial embedding to task-specific embedding (Rothe et al., 2016; Madhyastha et al., 2016), using definitions of OOV words from auxiliary data (Long et al., 2016; Bahdanau et al., 2017), and tracking contexts to build/update representations (Henaff et al., 2016; Kobayashi et al., 2017; Ji et al., 2017; Zhao et al., 2018).

Meanwhile, there have been efforts in representing words by utilizing character-level (Zhang et al., 2015; Ling et al., 2015a,b; Kim et al., 2016; Gimpel and Livescu, 2016) or subword-level representations (Sennrich et al., 2016; Bojanowski et al., 2017). To leverage the advantages in character and (sub)word level representation, some previous work combine (sub)word- and character-level representations (Santos and Zadrozny, 2014; dos Santos et al., 2015; Yu et al., 2017) or develop hybrid word/subword-character architectures (Chung et al., 2016; Luong and Manning, 2016; Pinter et al., 2017; Bahdanau et al., 2017; Matthews et al., 2018; Li et al., 2018). However, all those approaches assume monolingual setting, which is different from ours.

4.2 Multilingual Setting

Addressing OOV problems in a multilingual setting is relatively under-explored, probably because most multilingual models use separate vocabularies (Jaffe, 2017; Platanios et al., 2018). While there is no direct precedent, previous work show that incorporating multilingual contexts can improve monolingual word embeddings (Zou et al., 2013; Andrew et al., 2013; Faruqui and Dyer, 2014; Lu et al., 2015; Ruder et al., 2017).

Madhyastha and España-Bonet (2017) increase the vocabulary size for statistical machine translation (SMT). Given an OOV source word, they generate a translation list in target language, and integrate this list into SMT system. Although they also generate translation list (similar with us), their approach is still in monolingual setting with SMT. Cotterell and Heigold (2017) train char-level taggers to predict morphological taggings for high/low resource languages jointly, alleviating OOV problems to some extent. In contrast, we focus on dealing with the OOV issue at subword level in the context of pre-trained BERT model.

5 Conclusion

We investigated two methods (i.e., joint mapping and mixture mapping) inspired by monolingual solutions to alleviate the OOV issue in multilingual settings. Experimental results on several benchmarks demonstrate the effectiveness of mixture mapping and the usefulness of bilingual information. To the best of our knowledge, this is

the first work to address and discuss OOV issues at the subword level in multilingual settings. Future work includes: investigating other embedding alignment methods such as Gromov-Wasserstein alignment (Alvarez-Melis and Jaakkola, 2018) upon more languages; investigating approaches to choose the subwords to be added dynamically.

Acknowledgments

We thank the anonymous reviewers for their encouraging and helpful feedback.

References

Gustavo Aguilar, Fahad AlGhamdi, Victor Soto, Mona Diab, Julia Hirschberg, and Thamar Solorio. 2018. Overview of the CALCS 2018 Shared Task: Named Entity Recognition on Code-switched Data. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 138–147, Melbourne, Australia.

David Alvarez-Melis and Tommi Jaakkola. 2018. Gromov-wasserstein alignment of word embedding spaces. In *Proceedings of the EMNLP*, pages 1881–1890, Brussels, Belgium.

Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *Proceedings of the ICML*, pages 1247–1255, Atlanta, GA.

Duygu Ataman and Marcello Federico. 2018. Compositional representation of morphologically-rich input for neural machine translation. In *Proceedings of the ACL*, pages 305–311, Melbourne, Australia.

Dzmitry Bahdanau, Tom Bosc, Stanisław Jastrzebski, Edward Grefenstette, Pascal Vincent, and Yoshua Bengio. 2017. Learning to compute word embeddings on the fly. *arXiv preprint arXiv:1706.00286*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (wmt18). In *Proceedings of the WMT*, pages 272–303, Belgium, Brussels.

Huadong Chen, Shujian Huang, David Chiang, Xinyu Dai, and Jiajun Chen. 2018. Combining character and word information in neural machine translation using a multi-level attention. In *Proceedings of the NAACL-HLT*, pages 1284–1293.

Sébastien Jean Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the ACL-IJCNLP*, pages 1–10, Beijing, China.

Zewei Chu, Hai Wang, Kevin Gimpel, and David McAllester. 2017. Broad context language modeling as reading comprehension. In *Proceedings of the EACL*, pages 52–57, Valencia, Spain.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word translation without parallel data. In *Proceedings of the ICLR*, Vancouver, Canada.

Ryan Cotterell and Georg Heigold. 2017. Cross-lingual character-level neural morphological tagging. In *Proceedings of the EMNLP*, pages 748–759, Copenhagen, Denmark.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the ACL*, pages 1832–1846.

Kai Fan, Bo Li, Fengming Zhou, and Jiayi Wang. 2018. " bilingual expert" can find translation errors. *arXiv preprint arXiv:1807.09433*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the EACL*, pages 462–471, Gothenburg, Sweden.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Proceedings of NAACL-HLT*, pages 1296–1306.

John Wieting Mohit Bansal Kevin Gimpel and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the NAACL-HLT*, pages 344–354, New Orleans, LA.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Alan Jaffe. 2017. Generating image descriptions using multilingual data. In *Proceedings of the WMT*, pages 458–464, Copenhagen, Denmark.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839.

Yongyos Kaewpitakkun, Kiyoaki Shirai, and Masnizah Mohd. 2014. Sentiment lexicon interpolation and polarity estimation of objective and out-of-vocabulary words to improve sentiment classification on microblogging. In *Proceedings of the PACLIC*, pages 204–213, Phuket, Thailand.

Rasoul Kaljahi, Jennifer Foster, Johann Roturier, Corentin Ribeyre, Teresa Lynn, and Joseph Le Roux. 2015. Foreebank: Syntactic analysis of customer support forums. In *Proceedings of the EMNLP*, pages 1341–1347.

Yeachan Kim, Kang-Min Kim, Ji-Min Lee, and SangKeun Lee. 2018. Learning to generate word representations using subword information. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2551–2561.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Sosuke Kobayashi, Naoaki Okazaki, and Kentaro Inui. 2017. A neural language model for dynamically representing the meanings of unknown words and entities in a discourse. *arXiv preprint arXiv:1709.01679*.

Prasanth Kolachina, Martin Riedl, and Chris Biemann. 2017. Replacing oov words for dependency parsing with distributional semantics. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 11–19.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

Bofang Li, Aleksandr Drozd, Tao Liu, and Xiaoyong Du. 2018. Subword-level composition functions for learning word embeddings. In *Proceedings of the Second Workshop on Subword/Character LEvel Models*, pages 38–48.

Xiaoqing Li, Jiajun Zhang, and Chengqing Zong. 2016. Towards zero unknown word in neural machine translation. In *IJCAI*, pages 2852–2858.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by summarizing long sequences. In *Proceedings of the ICLR*, Vancouver, Canada.

Teng Long, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2016. Leveraging lexical resources for learning entity embeddings in multi-relational data. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 112.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1054–1063.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 11–19.

Pranava Swaroop Madhyastha, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Mapping unseen words to task-trained embedding spaces. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 100–110.

Pranava Swaroop Madhyastha and Cristina España-Bonet. 2017. Learning bilingual projections of embeddings for vocabulary expansion in machine translation. In *Proceedings of the RepL4NLP*, pages 139–145.

Austin Matthews, Graham Neubig, and Chris Dyer. 2018. Using morphological knowledge in open-vocabulary neural language models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1435–1445.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 92–97.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554.

Nanyun Peng and Mark Dredze. 2017a. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 91–100.

Nanyun Peng and Mark Dredze. 2017b. Supplementary results for named entity recognition on chinese social media with an updated dataset. Technical report.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the ACL*, pages 1756–1765, Vancouver, Canada.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 2227–2237.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 102–112.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 412–418.

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. 2018. Contextual parameter generation for universal neural machine translation. In *Proceedings of the EMNLP*, pages 425–435, Brussels, Belgium.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 157–163.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Majid Razmara, Maryam Siahbani, Reza Haffari, and Anoop Sarkar. 2013. Graph propagation for paraphrasing out-of-vocabulary words in statistical machine translation. In *Proceedings of the ACL*, pages 1105–1115, Sofia, Bulgaria.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 767–777.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. 2017. A survey of cross-lingual word embedding models. *arXiv preprint arXiv:1706.04902*.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the ACL*, pages 1715–1725, Berlin, Germany.

Anders Søgaard and Anders Johannsen. 2012. Robust learning in random subspaces: Equipping nlp for oov effects. *Proceedings of COLING 2012: Posters*, pages 1171–1180.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the ACL*, pages 778–788, Melbourne, Australia.

Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramón Astudillo, and André FT Martins. 2018. Findings of the wmt 2018 shared task on quality estimation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 689–709.

Karl Stratos. 2017. A sub-character architecture for korean language processing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 721–726.

Jeremie Tafforeau, Thierry Artieres, Benoit Favre, and Frederic Bechet. 2015. Adapting lexical representation and oov handling from written to spoken language with word embedding. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Julia M Taylor, Victor Raskin, and Christian F Hempelmann. 2011. Towards computational guessing of unknown word meanings: The ontological semantic approach. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33.

Changhan Wang, Kyunghyun Cho, and Douwe Kiela. 2018. Code-switched named entity recognition with embedding attention. In *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, pages 154–158.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Chao Xing, Dong Wang, Chao Liu, and Yiye Lin. 2015. Normalized word embedding and orthogonal transform for bilingual word translation. In *Proceedings of the NAACL-HLT*, pages 1006–1011, Denver, CO.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2017. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*.

Jinxing Yu, Xun Jian, Hao Xin, and Yangqiu Song. 2017. Joint embeddings of chinese words, characters, and fine-grained subcharacter components. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 286–291.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Zhuosheng Zhang, Yafang Huang, and Hai Zhao. 2018. Subword-augmented embedding for cloze reading comprehension. In *Proceedings of the COLING*, pages 1802–1814, Santa Fe, NM.

Yang Zhao, Jiajun Zhang, Zhongjun He, Chengqing Zong, and Hua Wu. 2018. Addressing troublesome words in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 391–400.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE ICCV*, pages 19–27, Santiago, Chile.

Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the EMNLP*, pages 1393–1398, Seattle, WA.

On the Relation between Position Information and Sentence Length in Neural Machine Translation

Masato Neishi
The University of Tokyo
neishi@tkl.iis.u-tokyo.ac.jp

Naoki Yoshinaga
Institute of Industrial Science,
the University of Tokyo
ynaga@iis.u-tokyo.ac.jp

Abstract

Long sentences have been one of the major challenges in neural machine translation (NMT). Although some approaches such as the attention mechanism have partially remedied the problem, we found that the current standard NMT model, Transformer, has difficulty in translating long sentences compared to the former standard, Recurrent Neural Network (RNN)-based model. One of the key differences of these NMT models is how the model handles position information which is essential to process sequential data. In this study, we focus on the position information type of NMT models, and hypothesize that relative position is better than absolute position. To examine the hypothesis, we propose RNN-Transformer which replaces positional encoding layer of Transformer by RNN, and then compare RNN-based model and four variants of Transformer. Experiments on ASPEC English-to-Japanese and WMT2014 English-to-German translation tasks demonstrate that relative position helps translating sentences longer than those in the training data. Further experiments on length-controlled training data reveal that absolute position actually causes overfitting to the sentence length.

1 Introduction

Sequence to sequence models for neural machine translation (NMT) are now utilized for various text generation tasks including automatic summarization (Chopra et al., 2016; Nallapati et al., 2016; Rush et al., 2015) and dialogue systems (Vinyals and Le, 2015; Shang et al., 2015); the models are required to take inputs of various length. Early studies on recurrent neural network (RNN)-based model analyze the translation quality with respect to the sentence length, and show that their models improve translations for long sentences, using the long short-term memory (LSTM) (Sutskever et al., 2014) or introducing the attention mechanism (Bahdanau et al., 2015; Luong et al., 2015). However, Koehn and Knowles (2017) report that even RNN-based model with the attention mechanism performs worse than phrase-based statistical machine translation (Koehn et al., 2007) in translating very long sentences, which challenges us to develop an NMT model that is robust to long sentences or more generally, variations in input length.

Have the recent advances in NMT achieved the robustness to the variations in input length? NMT has been advancing by upgrading the model architecture: RNN-based model (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015) followed by convolutional neural network (CNN)-based model (Kalchbrenner et al., 2016; Gehring et al., 2017) and attention-based model (Vaswani et al., 2017) called Transformer (§ 2). Transformer is the de facto standard NMT model today for its better performance compared to the former standard RNN-based model. We thus came up with a question whether Transformer have acquired the robustness to the variations in input length.

On the length of input sentence(s), the key difference between existing NMT models is how they incorporate information on word positions in the input. RNN or CNN-based NMT captures *relative* positions which stem from sequential operation of RNN or convolution operation of CNN. On the other hand, position embeddings or positional encodings (vector representations of positions) are used to handle *absolute* positions in Transformer. Gehring et al. (2017) integrate position embeddings, which are induced together with the other model parameters, into the CNN-based model, and showed that absolute position is still beneficial for their model in addition to the relative position captured by CNN. By contrast, Transformer only em-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 328–338
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

ploys positional encodings, which give fixed vectors to positions using sine and cosine functions.

In this study, we suspect that these differences in position information types of the models have an impact on the accuracy of translating long sentences, and investigate the impact of position information on translating long sentences to realize an NMT model that is robust to variations in input length. We reveal that RNN-based model (relative position) is better than Transformer with positional encodings (absolute position) in translating longer sentences than those in the training data (§ 5.2). Motivated from this result, we propose a simple modification to Transformer, using RNN as relative positional encoder (§ 4).

Whereas RNN and CNN-based models are inseparable from relative position inside of RNN or CNN, Transformer allows us to change the position information type. We therefore compare the RNN-based model and four variants of Transformer: vanilla Transformer, the modified Transformer using self-attention with relative positional encodings (Shaw et al., 2018), our modified Transformer with RNN instead of positional encoding layer, and a mixture of the last two models (§ 5). On ASPEC English-to-Japanese and WMT2014 English-to-German translation tasks, we show that relative information improves Transformer to be more robust to variations in input length.

Our contribution is as follows:

- We identified a defect in Transformer. Use of absolute position makes it difficult to translate very long sentences.
- We proposed a simple method to incorporate relative position into Transformer; it gives an additive improvement to the existing model by Shaw et al. (2018) which also incorporates relative position.
- We revealed the overfitting property of Transformer to both short and long sentences.

2 Related Work

Early studies on NMT, at that time RNN-based model, analyze the translation quality in terms of sentence length (Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015), and a few studies shed light on the details. Shi et al. (2016) examine why RNN-based model generates translations of the right length without special mechanism for the length, and report how LSTM regulates the output length. Koehn and Knowles (2017) reveal that RNN-based model has lower translation quality on very long sentences. Although researchers have proposed various new NMT architecture, they usually evaluate their models only in terms of the overall translation quality and rarely mention how the translation has changed (Gehring et al., 2017; Kalchbrenner et al., 2016; Vaswani et al., 2017). Only a few studies do the analysis on the translation quality in terms of sentence length (Elbayad et al., 2018; Zhang et al., 2019). The robustness of the recent NMT models on very long sentences remains to be assessed.

What we focus on in this study is the word position information which will closely relate to the decodable sentence length. Relative information has been implicitly used in the models using RNN or CNN. Gehring et al. (2017) introduce position embeddings which represent absolute position information to their CNN-based model. Sukhbaatar et al. (2015) introduce another absolute position information, positional encodings, which need no parameter training, and Vaswani et al. (2017) adopt them in their model, Transformer, which has neither RNN nor CNN.

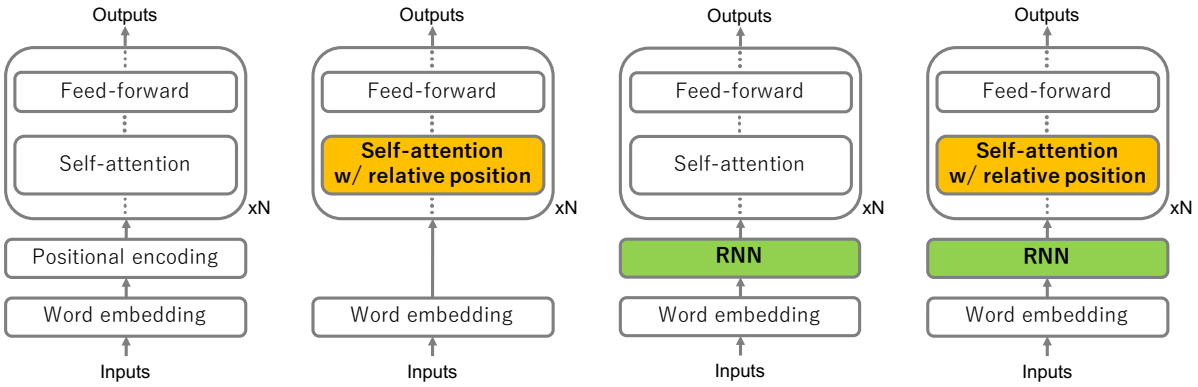
Recently, Shaw et al. (2018) propose to incorporate relative position into Transformer by modifying the self-attention layer while removing positional encodings. Lei et al. (2018) propose a fast RNN named Simple Recurrent Units (SRU) and replace the feed-forward layers of Transformer by SRU considering that recurrent process would better capture sequential information. Although both approaches succeeded in improving BLEU score, the researchers did not report in what respect the models improved the translation.

Chen et al. (2018) propose a RNN-based model, RNMT+, which is based on stacked LSTMs and incorporates some components from Transformer such as layer normalization and multi-head attention. On the other hand, our model is based on Transformer and incorporates RNN into Transformer.

3 Preliminaries

3.1 Transformer

Transformer (Vaswani et al., 2017) is a sequence to sequence model that has an encoder to process and represent input sequence and a decoder to generate output sequence from the encoder outputs. Both the encoder and decoder have a word embedding layer, a positional encoding layer, and



(a) Transformer (Vaswani et al., 2017)

(b) Rel-Transformer (Shaw et al., 2018)

(c) RNN-Transformer Our proposed

(d) RR-Transformer (b) + Our proposed

Figure 1: The architectures of all the Transformer-based models we compare in this study; for simplicity, we show the encoder architectures here since the same modification is applied to their decoders.

stacked encoder/decoder layers. The encoder architecture is shown in Figure 1a.

Word embedding layers encode input words into continuous low-dimension vectors, followed by positional encoding layers that add position information to them. Encoder/decoder layers consist of a few sub-layers, self-attention layer, attention layer (decoder only) and feed-forward layer, with layer normalization (Ba et al., 2016) for each. Both self-attention layer and attention layer employ the same architecture, and we explain the details in § 3.3. Feed-forward layer consists of two linear transformations with a ReLU activation in between. As for the decoder, a linear transformation and a softmax function follow the stacked layers to calculate probabilities of words to output.

Figure 1 illustrates the architectures of all the Transformer-based models we compare in this study including our porposed model which will be introduced in § 4. The model in Shaw et al. (2018) modifies the self-attention layer (§ 3.3).

3.2 Word Position Information

Transformer has positional encoding layers which follow the word embedding layers and capture absolute position. The process of positional encoding layer is to add positional encodings (position vectors) to input word embeddings. The positional encodings are generated using sinusoids of varying frequencies, which is designed to allow the model to attend to relative positions from the periodicity of positional encodings (sinusoids). This is in contrast to the position embeddings (Gehring et al., 2017), a learned position vectors, which are not meant to attend to relative positions. Vaswani et al. (2017) report that both approaches produced nearly identical results in their experiments, and also mentioned that the model with positional encodings may handle longer inputs in testing than those in training, which implies that absolute position approach might have problems at this point.[1]

3.3 Self-attention with Relative Position

Some studies modify Transformer to consider relative position instead of absolute position. Shaw et al. (2018) propose an extension of self-attention mechanism which handles relative position inside in order to incorporate relative position into Transformer. We hereafter refer to their model as Rel-Transformer. In what follows, we explain the self-attention mechanism and their extension.

Self-attention is a special case of general attention mechanism, which uses three elements called query, key and value. The basic idea is to compute weighted sum of values where the weights are computed using the query and keys. Each weight represents how much attention is paid to the corresponding value. In the case of self-attention, the input set of vectors behaves as all of the three elements (query, key and value) using three different transformations. When taking a sentence as input, it is processed as a set in the self-attention.

Self-attention operation is to compute output sequence $z = (z_1, ..., z_n)$ out of input sequence $x = (x_1, ..., x_n)$, where both sequences have the same langth n and $x_i \in \mathbb{R}^{d_x}, z_i \in \mathbb{R}^{d_z}$. The output

[1] Our preliminary experiment confirmed that positional encodings perform better for longer sentences than those in the training data, while position embeddings perform slightly better for the other length.

element z_i is computed as follows.

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V) \quad (1)$$

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}} \quad (2)$$

$$e_{ij} = \frac{x_i W^Q (x_j W^K)^T}{\sqrt{d_z}}, \quad (3)$$

where $W^Q, W^K, W^V \in \mathbb{R}^{d_x \times d_z}$ are the matrices that transform input elements into querys, keys, and values, respectively.

The extension proposed by Shaw et al. (2018) adds only two terms to the original self-attention: the relative position vectors $w^K_{j-i}, w^V_{j-i} \in \mathbb{R}^{d_z}$.

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + w^V_{j-i}) \quad (4)$$

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}} \quad (5)$$

$$e_{ij} = \frac{x_i W^Q (x_j W^K + w^K_{j-i})^T}{\sqrt{d_z}}, \quad (6)$$

Note that when using the relative position vectors, the input is processed as a directed graph instead of a set. Maximum distance k is employed to clip the relative distance within a certain distance so that the value of relative distance is limited as $-k < j - i < k$.

4 RNN as a Relative Positional Encoding

The approach by Shaw et al. (2018) is not the only way to incorporate relative position into Transformer. Lei et al. (2018) replace feed-forward layers by their proposed SRU which also incorporates relative position. Both approaches modify the encoder and decoder layers that are repeatedly stacked, which means their models handle position information multiple times. However, the original Transformer does only once at the positional encoding layer which locates shallow layer of the deep layered network.

To conduct a clear comparison of the position information types, we propose another simple method that replaces the positional encoding layer of Transformer by RNN. As the RNN has the nature to handle a sequence using relative position information, it can be used not only as a main processing unit of RNN-based model, but also as a relative positional encoder. While Lei et al. (2018) also employ RNN, they use position embeddings. Our approach is a pure replacement of position information type for Transformer.

In the original Transformer, the positional encoding layer adds the i-th position vector $pe(i) \in \mathbb{R}^{d_{wv}}$ to the i-th input word vector $wv_i \in \mathbb{R}^{d_{wv}}$ and outputs the position informed word vector $wv'_i \in \mathbb{R}^{d_{wv}}$:

$$wv'_i = wv_i + pe(i) \quad (7)$$

In our approach, we adopt RNN, specifically GRU (Cho et al., 2014) in this study, as a relative positional encoder. GRU computes its output or its i-th time hidden state $h_i \in \mathbb{R}^{d_{wv}}$ given the input word vector wv_i and the previous hidden state $h_{i-1} \in \mathbb{R}^{d_{wv}}$, and we take h_i as the position informed word vector wv'_i:

$$h_i = \mathrm{GRU}(wv_i, h_{i-1}) \quad (8)$$

$$wv'_i = h_i \quad (9)$$

Although LSTM (Hochreiter and Schmidhuber, 1997) is more often used as an RNN module in RNN-based models, we employed GRU which has less parameters. This is because, in our approach, RNN is just a positional encoder which we do not expect to work more, even though it can. We refer to our proposed model as RNN-Transformer.

We also consider the mixture of Shaw et al. (2018) and our method to investigate whether the two methods of considering relative position have additive improvements. Although both methods are intended to incorporate relative position into Transformer, they modify different parts of Transformer. By combining both, we can see either of modification suffices to incorporate relative position. We refer to this model as RR-Transformer.

5 Experiments

We conduct two experiments to evaluate our modification to Transformer and to investigate the impact of using relative position in NMT models. The first experiment is a basic translation experiment which uses all the training data. We carry out analysis on the translations generated by the NMT models in terms of sentence length, especially focusing on long sentences. In the second experiment, we control the training data by the sentence length so that the NMT models are trained only on sentences with lengths in a certain range. We also analyze the result in terms of sentence length, focusing on the short sentences.

	Train (orig.)	Dev	Test
ASPEC (En-Ja)	1,166,725 (3,008,500)	1790	1812
WMT2014 (En-De)	3,661,035 (4,468,840)	3000	2737

Table 1: Number of sentence pairs in the preprocessed corpus.

5.1 Setup

Dataset and Preprocess: We perform a series of experiments on English-to-Japanese and English-to-German translation tasks. For English-to-Japanese translation task, we exploit ASPEC (Nakazawa et al., 2016), a parallel corpus compiled from abstract sections of scientific papers. For English-to-German translation task, we exploit a dataset in WMT2014, which is one of the most common dataset for translation task.

For ASPEC English-to-Japanese data, we used scripts of Moses toolkit[2](ver. 2.2.1) (Koehn et al., 2007) for English tokenization and truecasing, and KyTea[3] (ver. 0.4.2) (Neubig et al., 2011) for Japanese segmentations. Following those word-level preprocess, we further applied SentencePiece (Kudo and Richardson, 2018) to segment texts down to subword level with shared vocabulary size of 16,000. Finally we selected the first 1,500,000 sentence pairs for the poor quality of the latter part, and filtered out sentence pairs with more than 49 subwords in either of the languages.

For WMT2014 English-to-German translation task, we used preprocessed data provided from the Stanford NLP Group,[4] and used newstest2013 and newstest2014 as development and test data, respectively. We also applied SentencePiece to this data to segment into subwords with shared vocabulary size of 40,000. We filtered out the sentence pairs in the same way as the ASPEC. Table 1 shows the number of sentence pairs of preprocessed data.

Figure 2 shows the distributions of the sentences plotted against the length of input sentence. Althought ASPEC data has slightly larger peak at sentence length of 20-29 subwords, both datasets have no big difference in length distributions. The training and test data have almost identical curves.

Model: We compare the following five NMT models:

[2] http://www.statmt.org/moses/

[3] http://www.phontron.com/kytea/

[4] https://nlp.stanford.edu/projects/nmt/

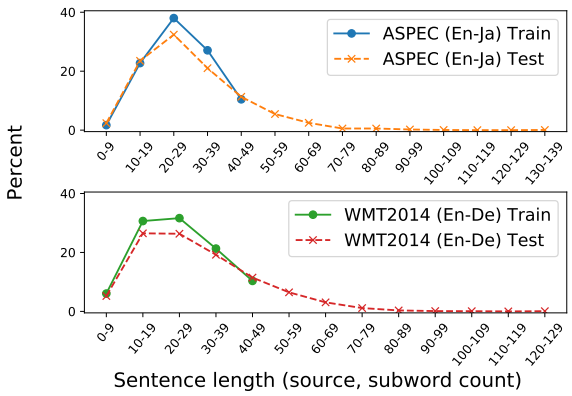


Figure 2: Sentence length ratio of preprocessed corpus.

RNN-NMT is a RNN-based NMT model with dot-attention and input-feeding (Luong et al., 2015). This model consists of four layered bi-directional LSTM for encoder and three layered uni-directional LSTM for decoder.

Transformer is a vanilla Transformer model (the base model in Vaswani et al. (2017)). This model consists of six-layered Transformer encoder and decoder.

Rel-Transformer is a modified version of Transformer by Shaw et al. (2018). Since the modifications do not increase the number of model parameter much, this model consists of the same number of encoder/decoder layers as **Transformer**, with the modified self-attention layer. We set the hyperparameter k, relative distance limit, to 16 following the base model in Shaw et al. (2018).

RNN-Transformer is another modified version of Transformer proposed in § 4. Because the replacement of the original positional encoding layer with RNN increases the number of model parameter, we employ uni-directional GRU as relative positional encoder for both encoder and decoder and reduced the number of decoder layer instead. Our RNN-Transformer model consists of six layered encoder and five layered decoder with one GRU layer for each.[5]

RR-Transformer is the mixture model of Rel-Transformer and RNN-transformer. With the same logic as Rel-Transformer, this model

[5] This configuration was chosen because it performed better than a model with five-layered encoder and six-layered decoder, and was comparable to five-layred encoder and decoder with bi-directional (instead of uni-directional) GRU for the relative position encoder in preliminary experiments.

	ASPEC (En-Ja)	WMT2014 (En-De)
RNN-NMT	70,521,476	107,409,476
Transformer	68,736,644	105,624,644
Rel-Transformer	68,787,332	105,675,332
RNN-Transformer	67,684,484	104,572,484
RR-Transformer	67,730,948	104,618,948

Table 2: Number of the model parameters.

consists of the same number of encoder and decoder layers as RNN-Transformer model, with the modified self-attention layer.

We implemented all the models using PyTorch[6] (ver. 0.4.1). Taking the base model of Transformer (Vaswani et al., 2017) which consists of six-layered encoder and decoder as a reference model, we built the other models to have almost the same number of model parameters for a fair comparison. For all models, we set word embedding dimension and model dimension (or hidden size for RNNs) to 512. For the Transformer-based models, we set feed-forward layer dimension to 2048, and the number of attention head to 8.

Table 2 shows the total number of model parameters for all the models in our implementation. The difference of the numbers by the datasets comes from the difference in vocabulary size.

Training: We used Adam optimizer (Kingma and Ba, 2015) with initial learning rate of 0.0001, and set dropout rate of 0.2 and gradient clipping value of 3.0. We adopted warm-up strategy (Vaswani et al., 2017) for fast convergence with warm-up step of 4k, and trained all the model for 300k steps. The mini-batch size was set to 128.

Evaluation: We performed greedy search for translation with the models, and evaluated the translation quality in terms of BLEU score (Papineni et al., 2002) using `multi-bleu.perl` in the Moses toolkit. We checked model's BLEU score on the development data at every 10k steps during the training, and took the best performing model for evaluation on the test data.

5.2 Long Sentence Translation

Table 3 shows the BLEU scores of the NMT models on the test data of ASPEC English-to-Japanese and WMT2014 English-to-German when using all the preprocessed training data for training. Table 4 lists the results of statistical significance test using bootstrapping of 10,000 samples. The evaluation is done on word-level, which means that we converted the outputs of NMT models from subword-level into word-level before scoring. On both datasets, Transformer outperforms RNN-NMT, and all of the three modified versions of Transformer outperform the Transformer. RNN-Transformer was comparable to Rel-Transformer, and RR-Transformer, the mixture of RNN-Transformer and Rel-Transformer, gives the best score.

	ASPEC (En-Ja)	WMT2014 (En-De) newstest2014
RNN-NMT	36.67	19.95
Transformer	38.44	21.00
Rel-Transformer	39.58	22.51
RNN-Transformer	39.17	22.35
RR-Transformer	**40.34**	**23.01**

Table 3: BLEU scores on test data.

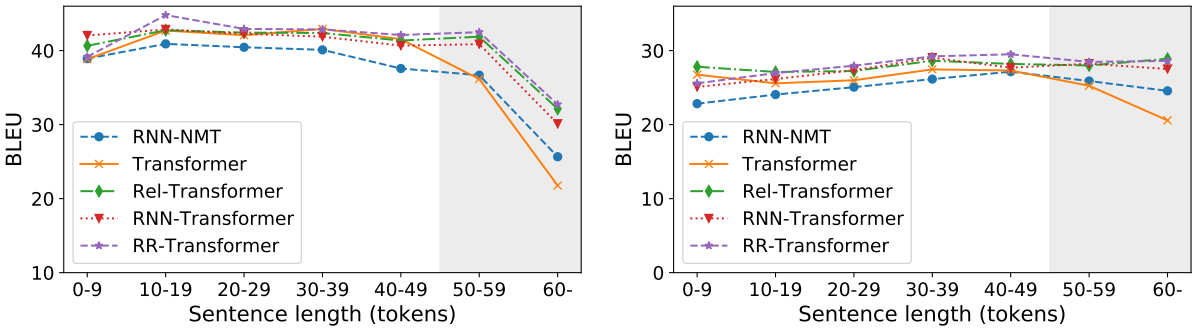
	RNN-NMT	Trans	Rel	RNN	RR
RNN-NMT		$<<$	$<<$	$<<$	$<<$
Trans	$>>$		$<<$	$<<$	$<<$
Rel	$>>$	$>>$		$\sim$	$<$
RNN	$>>$	$>>$	$\sim$		$<<$
RR	$>>$	$>>$	$>>$	$>>$	

Table 4: Results of statistical significance test on ASPEC English-to-Japanese (lower-left) and WMT2014 English-to-German (upper-right): "$>>$" or "$<<$" means $p < 0.01$, "$>$" or "$<$" means $p < 0.05$ and "$\sim$" means $p \geq 0.05$.

In order to see the capability of translating long sentences of the models, we split the test data into different bins according to the length of input sentences, and then calculated BLEU scores on each bin. The following evaluation uses the raw subword-level outputs of the models since the sentence length is based on subwords.

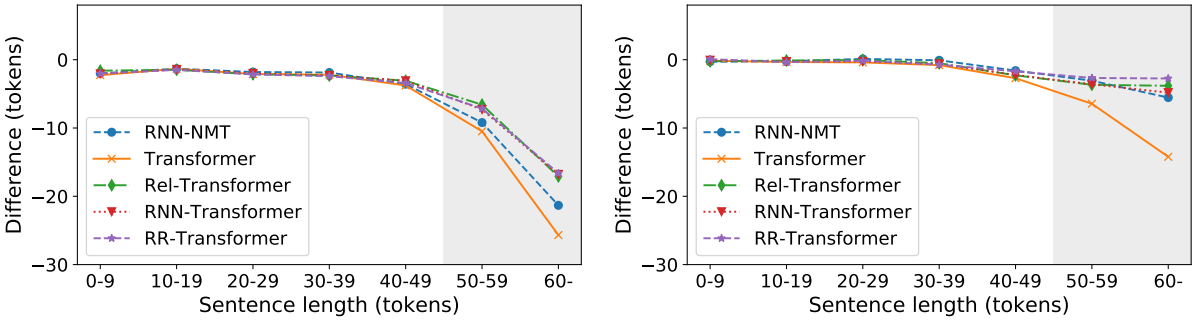
Figure 3a and 3b show the BLEU scores on the split test data of ASPEC English-to-Japanese and WMT2014 English-to-German, respectively. The BLEU score of Transformer, the only model that uses absolute position, more sharply drops than the BLEU scores of the other models at the input length of 50-59, which is outside of the length range of the training data. As for the input length of 60-, Transformer performs the worst among all the models. These results indicate that relative position works better than absolute position in translating sentences longer than those of the training data. Meanwhile, for the lengths with enough

[6] https://pytorch.org/



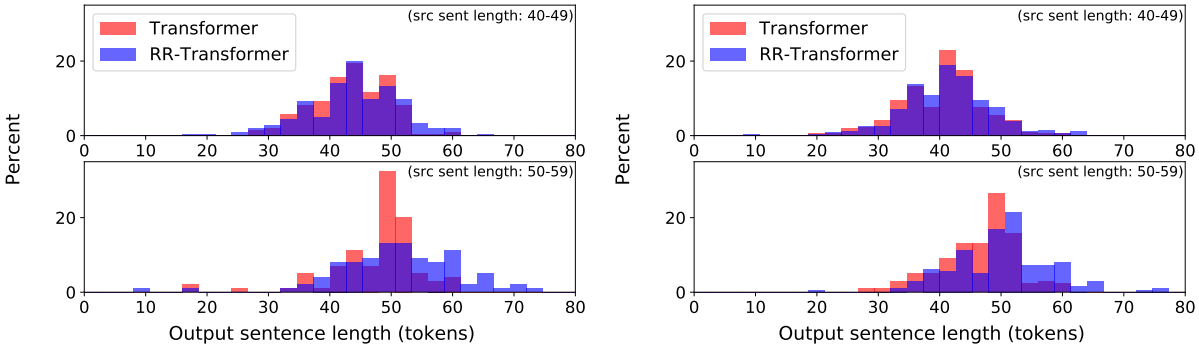
(a) ASPEC English-to-Japanese (b) WMT2014 English-to-German

Figure 3: BLEU scores on test data split by the sentence length (no training data in the gray-colored area).



(a) ASPEC English-to-Japanese (b) WMT2014 English-to-German

Figure 4: Averaged difference of sentence length between NMT model's output and the reference translation (no training data in the gray-colored area).



(a) ASPEC English-to-Japanese (b) WMT2014 English-to-German

Figure 5: Distributions of output sentence length of Transformer and RR-Transformer.

amount of training data, both position information types seem to work almost equally. On WMT2014 English-to-German, all the models except Transformer successfully keep as good performance in 50-59 and 60- bins as the other bins.

To figure out the effect of position information on the ability of the models to generate output of proper length, we look into the difference of sentence length between the model's output and the reference translation. Figure 4a and 4b show the averaged differences plotted against the input sentence length on both language pairs. We can observe that all the models tend to output shorter sentence than the reference. However, Transformer shows the largest drop at the input length of 50-59 again among all the models, which is even more than RNN-NMT. The difference between Transformer and RNN-Transformer indicates the advantage of relative position against absolute position, while the difference between the three modified Transformer-based models and RNN-NMT indicates the structural advantage of Transformer to RNN-based model in generating translations with appropriate lengths.

	min len.	max len.	# of sentences	# of tokens
Short	2	26	555,922	10,392,775
Middle	26	34	350,176	10,392,797
Long	34	49	260,626	10,392,729

(a) ASPEC English-to-Japanese

	min len.	max len.	# of sentences	# of tokens
Short	1	24	1,878,354	29,841,533
Middle	24	34	1,041,794	29,841,531
Long	34	49	740,887	29,841,519

(b) WMT2014 English-to-German

Table 5: Statistics of the split training data.

The above result that the models tend to output shorter sentences suggests that the models may have a limit in the range of output length. To confirm this possibility, we look into the distributions of the model's output length. Figure 5a and 5b show distributions of output length of Transformer and RR-Transformer for the input length of 40-49 (length within the training data) and 50-59 (length outside of the training data). For the input length of 40-49, the distributions of both models are flat and have no big difference. For the input length of 50-59, on the other hand, we can see a sharp peak in the distribution of Transformer in which most of the values distribute around 50 tokens or less. These results indicate that Transformer tends to overfit to a range of length of input sentences.

5.3 Length-Controlled Training Data

The above experiments focus on trainslation of long sentences, or, strictly speaking, sentences longer than those in the training data. With the use of absolute position, it is no surprise that the model fails to handle longer sentences since those sentences demand the model to handle the position vectors which are never seen during training.

In this section, we focus on short sentences to investigate whether Transformer overfits to the length of input sentences in the training data. Note that position vectors of small numbers are included in long sentences. If the problem is only unseen position vectors, then the model shall be able to handle short sentences because short sentences do not include any unseen position numbers.

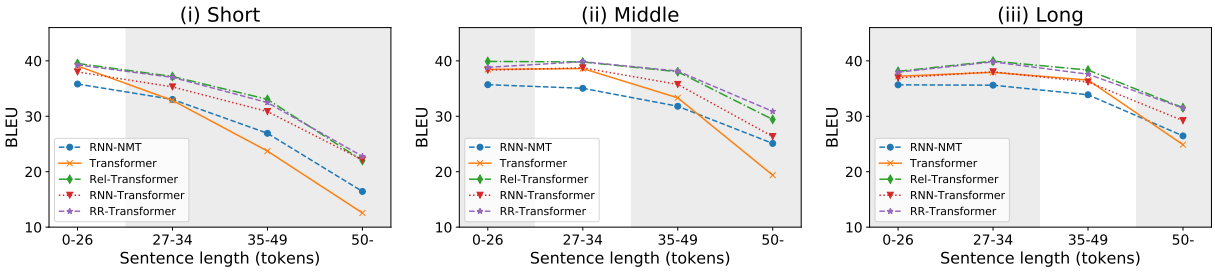
To figure out how the NMT models behave on sentences shorter than those in the training data, we conduct another experiment in which the length of the training data is controlled. We split the training data of both ASPEC English-to-Japanese and WMT2014 English-to-German into three portions according to the length of input sentences so that each of them has almost the same number of tokens. We then trained the five NMT models on each of the three training data. We hereafter refer to these three length-controlled training data as Short, Middle and Long. The statistics of these data is summarized in Table 5a and 5b.

To see how the translation quality changes between inside and outside of the length within the training data, we split the test data with respect to the lengths of split training data. Figure 6a and 6b show the BLEU scores on all the three training data of both language pairs. Transformer shows the worst performance among the four Transformer-based models on the sentences longer than those in the training data for any controlled length. However, on the shorter sentences than those in the training data, RNN-Transformer scores almost the same as Transformer on the Middle and Long training data of ASPEC English-to-Japanese and also shows a larger drop than RNN-NMT at length of -24 on the Long training data of WMT2014 English-to-German. This implies that our proposed method to replace absolute positional encoding layer by RNN does not work well in translating shorter sentences.

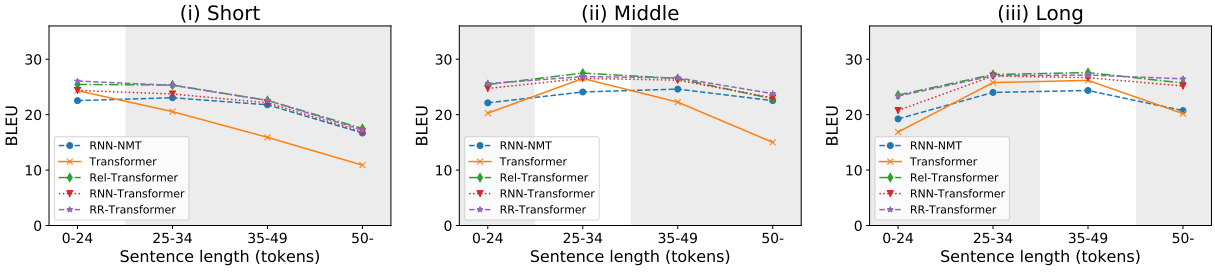
We can also see that Rel-Transformer and RR-Transformer are quite competitive across all the situations. This suggests that one Transformer decoder layer and two GRUs contribute almost equally to the translation quality.

Figure 7a and 7b show the averaged difference of length between NMT model's output and the reference translation on Long training data of both datasets.[7] These figures indicate that Transformer and RNN-Transformer tend to generate inappropriately long sentences in translating much shorter sentences than those in the training data. As mentioned above, when translating short sentences, there is no unseen positions in Transformer, while there is no concrete position representation in RNN-Transformer; the above results suggest that these two models overfit to the (longer) length of input sentences. In contrast, the result of Rel-

[7] Note that Figure 7a and 7b use different x-axis scale from Figure 6a and 6b in order to show the difference clearly.

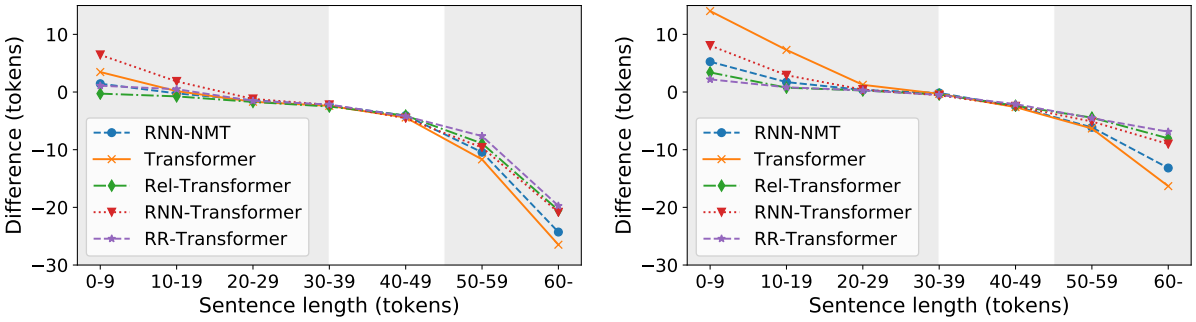


(a) ASPEC English-to-Japanese



(b) WMT2014 English-to-German

Figure 6: BLEU scores of models trained on three length-controlled training data on test data split in the same way as the training data (almost no training data in the gray-colored area).



(a) ASPEC English-to-Japanese

(b) WMT2014 English-to-German

Figure 7: Averaged difference of sentence length between NMT model's output translation and reference translation (almost no training data in the gray-colored area).

Transformer and RR-Transformer indicates that self-attention with relative position prevents this overfitting.

6 Conclusions

In this paper, we examined the relation between position information and the length of input sentences by comparing absolute position and relative position using RNN-based model and variations of Transformer models. Experiments on all the preprocessed training data revealed the crucial weakness of the original Transformer, which uses absolute position, in translating sentences longer than those of the training data. We also confirmed that incorporating relative position into Transformer helps to handle those long sentences and improves the translation quality. Another experiment on the length-controlled training data revealed that absolute position of Transformer causes overfitting to the input sentence length. To conclude, all the experiments suggest to use relative position and not to use absolute position.

Considering that the available data is not balanced in terms of the sentence length in practice, preventing the overfitting is useful for building a practical NMT system.

Acknowledgments

We deeply thank Satoshi Tohda for proofreading the draft of our paper. This work was partially supported by JST CREST Grant Number JPMJCR19A4, Japan. This research was also partially supported by NII CRIS Contract Research 2019.

References

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the third International Conference on Learning Representations (ICLR)*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 76–86.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 93–98.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2018. Pervasive attention: 2D convolutional neural networks for sequence-to-sequence prediction. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL)*, pages 97–107.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1243–1252.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR*, abs/1610.10099.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the third International Conference on Learning Representations (ICLR)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL): Demo and Poster Sessions*, pages 177–180.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Taku. Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, pages 66–71.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4470–4481.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian scientific paper excerpt corpus. In *Proceedings of the tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2204–2208.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Guçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 280–290.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT), Short Papers*, pages 529–533.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015*

Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 379–389.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 1577–1586.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Short Papers*, pages 464–468.

Xing Shi, Kevin Knight, and Deniz Yuret. 2016. Why neural translations are the right length. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2278–2282.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems (NIPS) 28*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS) 27*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS) 30*, pages 5998–6008.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *Proceedings of Deep Learning Workshop held at the 31st International Conference on Machine Learning (ICML)*.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4334–4343.

Word Recognition, Competition, and Activation in a Model of Visually Grounded Speech

William N. Havard[1,2], **Jean-Pierre Chevrot**[2], **Laurent Besacier**[1]

[1] LIG, Univ. Grenoble Alpes, CNRS, Grenoble INP, 38000 Grenoble, France
[2] LIDILEM, Univ. Grenoble Alpes, 38000 Grenoble, France
first-name.lastname@univ-grenoble-alpes.fr

Abstract

In this paper, we study how word-like units are represented and activated in a recurrent neural model of visually grounded speech. The model used in our experiments is trained to project an image and its spoken description in a common representation space. We show that a recurrent model trained on spoken sentences implicitly segments its input into word-like units and reliably maps them to their correct visual referents. We introduce a methodology originating from linguistics to analyse the representation learned by neural networks – the gating paradigm – and show that the correct representation of a word is only activated if the network has access to first phoneme of the target word, suggesting that the network does not rely on a global acoustic pattern. Furthermore, we find out that not all speech frames (MFCC vectors in our case) play an equal role in the final encoded representation of a given word, but that some frames have a crucial effect on it. Finally, we suggest that word representation could be activated through a process of lexical competition.

1 Introduction

Neural models of Visually Grounded Speech (VGS) sparked interest in linguists and cognitive scientists as they are able to incorporate multiple modalities in a single network and allow the analysis of complex interactions between them. Analysing these models does not only help to understand their technological limitations, but may also yield insight on the cognitive processes at work in humans (Dupoux, 2018) who learn from contextually grounded speech utterances (either visually, haptically, socially, etc.). This is with this idea in mind that one of the first computational model of visually grounded word acquisition was introduced by Roy and Pentland (2002). More recently, Harwath et al. (2016) and Chrupała et al. (2017) were among the first to propose neural models integrating these two modalities.

While Chrupała et al. (2017) and Alishahi et al. (2017) focused on analysing speech representations learnt by speech-image neural models from a phonological and semantic point of view, the present work focuses on lexical acquisition and the way speech utterances are segmented into lexical units by a neural model.

More precisely, we aim at understanding how word-like units are processed by a VGS architecture. First, we study if such models are robust to isolated word stimuli. As such networks are trained on raw speech utterances, robustness to isolated word stimuli would indicate that a segmentation process was implicitly carried out at training time. We also explore which factors influence the most such *word recognition*. In a second step, to better understand how individual words are activated by the network, we adapt the gating paradigm initially introduced to study human word recognition (Grosjean, 1980) where our neural model is inputted with speech segments of increasing duration (*word activation*). Finally, as some linguistic models assume that the first phoneme of a target word activates all the words starting by the same phoneme, we investigate if such a pattern holds true for our neural model as well (*word competition*). As far as we know, no other study has examined patterns of word recognition, activation and competition in models of VGS.

This paper is organised as follows: section 2 presents related works and section 3 details our experimental material (data and model). Our contributions follow in section 4 (word recognition), section 5 (word activation) and section 6 (word competition). Section 7 concludes this work.

2 Related Work

In this section we explore what is known about word recognition in humans. We then review recent works related to the representation of lan-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 339–348
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

guage in VGS models. A few words are also said about modified inputs and adversarial attacks as they are related to the analysis methodology used in part of this work.

2.1 Word Recognition in Humans

Many psycholinguistic models try to account for how words are activated and recognised from fluent speech. The process of word recognition *"requires matching the spoken input with mental representations associated with word candidates"* (Dahan and Magnuson, 2006). One of the first model trying to account for how humans recognise and extract words from fluent speech is the COHORT model by Marslen-Wilson and Welsh (1978). In this model, word recognition proceeds in 3 steps: *access*, *selection* and *integration*. *Access* denotes the process by which a set of words (a cohort) becomes activated if their onsets are consistent with the perceived spoken input. As soon as a word form becomes inconsistent with the spoken input, it is removed from the initial cohort (*selection* phase). A word is deemed recognised as soon it is the last one standing in the cohort. *Integration* consists in checking if the word's syntactic and semantic properties are consistent with the rest of the utterance. However, COHORT supposes a full match between the perceived input and the word forms and does not account for word frequency in the access phase. REVISED COHORT (Marslen-Wilson, 1987) later relaxed the constraints on the cohort formation to take into account these facts. There is no active competition *per se* between words in the COHORT model. That is, the strength of activation of a word does not depend on the value of the activation of the other words, but only on how well the internalised word form matches the perceived spoken input. TRACE (McClelland and Elman, 1986) is a connectionist model of spoken word recognition consisting of three layers of nodes, where each layer represents a particular linguistic unit (feature, phoneme and word). Layers are linked by exitatory connections (e.g. fricative feature node would activate /f/ phoneme node which would, in turn, activate words starting with this sound), and nodes within a layer are linked by inhibitory connections, thus inducing a real competition between activated words. Contrary to the COHORT model which does not allow words embedded in longer words to be activated, TRACE allows such activation. SHORTLIST (Norris, 1994) is another model which builds upon COHORT and TRACE by taking into consideration other features such as word stress.[1]

To sum up, models of spoken word recognition consider that a set of words matching to a certain extent the spoken input is simultaneously activated and these models involve at some point a form of competition between the set of activated words before reaching the stage of recognition.

2.2 Computational Models of VGS

Roy and Pentland (2002) were among the first to propose a computational model, known as CELL, that integrates both speech and vision to study child language acquisition. However, CELL required both speech and images to be pre-processed, where canonical shapes were first extracted from images and further represented as histograms; and speech was discretised into phonemes. More recently, CNN-based VGS models (Harwath et al., 2016, 2018; Kamper et al., 2019) and RNN-based VGS models (Chrupała et al., 2017) which do not require speech to be discretised into sub-units were introduced. Chrupała et al. (2017) investigated how RNN-based models encode language, and showed such models tend to encode semantic information in higher layers, while form is better encoded in lower layers. Alishahi et al. (2017) studied if such models capture phonological information and showed that some layers do capture such information more accuratly than others. Kádár et al. (2017) introduced *omission scores* to interpret the contribution of individual tokens in text-based VGS models. More recently, Havard et al. (2019) studied the behaviour of attention in RNN-based VGS models and showed that these models tend to focus on nouns and could display language-specific patterns, such as focusing on particules when prompted with Japanese. Recently, Harwath et al. (2018) showed that CNN-based models could reliably map word-like units to their visual referents, and Harwath and Glass (2019) showed such networks were sensitive to diphone transitions and that these were useful for the purpose of word recognition. However, none of the aforementioned works studied the process by which words are recognised and activated. This present work aims at bridging what is known about word activation

[1]For a review of spoken word recognition model, reader can consult Dahan and Magnuson (2006) and Weber and Scharenborg (2012)

and recognition in humans and the computations at work in VGS models.

2.3 Modified Inputs and Adversarial Attacks

As will be shown later, the gating method used in this article modifies the input stimulus to better understand the behaviour of the neural model.

We can draw a parallel with approaches recently introduced to show the vulnerability of deep networks to strategically modified samples (adversarial examples) and to detect their over-sensitivity and over-stability points. It was shown that imperceptible perturbations can fool the neural models to give false predictions. Inspired by the researches for images (Su et al., 2019), efforts on attacking neural networks for NLP applications emerged recently (see Zhang et al. (2019) for a survey). However, while a lot of references can be found for textual adversarial examples, fewer papers addressed adversarial attacks for speech (we can however mention the work of Wu et al. (2014) addressing spoofing attacks in speaker verification and of Carlini and Wagner (2018) attacking *DeepSpeech* end-to-end ASR system).

3 Experimental Settings

3.1 Model Type

Even though the methodologies developed in this work could also be applied to CNN-based VGS models, the present work will solely focus on the analysis of the representations learned by a RNN-based VGS model. Indeed, from a cognitive perspective, RNN-based models are more realistic than CNN-based models as the speech signal – or in our case, a sequence of MFCC vectors – is sequentially processed from left-to-right, whereas in CNN-based models the network processes multiple frames at the same time. This will thus allow us to explore if RNN-based models display human-like behaviour or not.

3.2 Model Architecture

The model we use for our experiment is based on that of Chrupała et al. (2017) and later modified by Havard et al. (2019). It is trained to solve an image retrieval task: given a speech query, the model should retrieve the closest matching image. The model consists of two parts: an image encoder and a speech encoder. The image encoder takes VGG-16 pre-calculated vectors as input instead of raw images. It consists of a dense layer which reduces the 4096 dimensional VGG-16 input vector into a 512 dimensional vector which is then L2 normalised. The speech encoder takes 13 Mel Frequency Cepstral Coefficients (MFCC) vectors instead of raw speech.[2] It consists of a convolutional layer (64 filters of length 6 and stride 3) followed by 5 stacked unidirectional GRU layers (Cho et al., 2014), with 512 units each. Two attention mechanisms (Bahdanau et al., 2015) are used: one after the 1^{st} recurrent layer and one after the 5^{th} recurrent layer. The final vector produced by the speech encoder corresponds to the dot product of the weighted vectors outputted by each attention mechanism. The model is trained to minimise the following triplet loss function as implemented by Chrupała et al. (2017):

$$\mathcal{L}(u,i,\alpha) = \sum_{u,i}\left(\sum_{u'}\max[0,\alpha + d(u,i) - d(u',i)] + \sum_{i'}\max[0,\alpha + d(u,i) - d(u,i')]\right) \quad (1)$$

The loss function encourages the network to minimise the cosine distance d between an image i and its corresponding spoken description u by a given margin α while maximising the distance between mismatching image/utterance pairs. For our experiments, we set $\alpha = 0.2$.

3.3 Data

The data set used for our experiments is based on MSCOCO (Lin et al., 2014). MSCOCO is a data set used to train computer vision systems, and features annotated images, each paired with 5 human written descriptions in English. MSCOCO's images where selected so that the images would contain instances of 80 possible object categories. We trained our model on the spoken extension introduced by Chrupała et al. (2017). This extension provides spoken version of the human written captions. It is worth mentioning that this extended data set features synthetic speech (female voice generated using Google's Text-To-Speech (TTS) system) and not real human speech.

[2] 12 mel frequency cepstral coefficients + log of total frame energy, vectors extracted every 10ms on a 25ms window

3.4 Model Training and Results

We trained our model for 15 epochs with Adam optimiser and an initial learning rate of 0.0002. The training set comprises 113,287 images with 5 spoken captions per image. Validation and test set comprise 5000 images each.[3] Model is evaluated in term of Recall@k (R@k) and median rank $\widetilde{r}$. That is, given a spoken query, which corresponds to a full utterance, we evaluate the model's ability to rank the unique paired image in the top k images. We obtain a $\widetilde{r}$ of 28.

Full results are shown in Table 1. Even though our results are lower than the original implementation by Chrupała et al. (2017), our model still performs far above chance level, showing it did learn how to map an image and its spoken description.

Model	R@1	R@5	R@10	$\widetilde{r}$
Synth. COCO	0.056	0.182	0.284	28

Table 1: Recall at 1, 5, and 10 results and median rank $\widetilde{r}$ on a speech-image retrieval task (test part of our datasets with 5k images). Chrupała et al. (2017) with RHN reports median rank $\widetilde{r} = 13$. Chance median rank $\widetilde{r}$ is 2500.5.

4 Word Recognition

Harwath et al. (2018) observed that CNN-based models can reliably map word-like units to their corresponding visual reference. Chrupała et al. (2017) and more recenlty Merkx et al. (2019) showed that RNN-based utterance embeddings contain information about individual words, but did not show for what type of words this behaviour holds true and if the model had learnt to map these individual words to their visual referents. Havard et al. (2019) showed that the attention mechanism of RNN-based VGS models tends to focus on the end of words that correspond to the main concept of the target image. This suggests that such models are able to isolate the target word forms from fluent speech and thus segment their inputs into sub-units. In the following experiment we test if a RNN-based VGS network can reliably map isolated word-like units to their visual referents and explore the factors that could influence such mapping.

[3]The train/dev/test correspond to those used in (Chrupała et al., 2017)

4.1 Isolated Word Mapping

We selected a set of 80 words corresponding to the name of 80 object categories in the MSCOCO data set.[4] We expect our model to be very efficient with the selected 80 words, as these are the main objects featured in MSCOCO. We generated speech signals for these 80 isolated words using Google's TTS system and then extracted MFCC features for each of the generated words. We evaluate the ability of the model to rank images containing an object instance corresponding to the target word among the first 10 images (P@10).[5] Contrary to (Chrupała et al., 2017) who uses Recall@k, we use Precision@k as there are several images that correspond to a single target word. It is to be noted that at training time, the network was only given full captions and not isolated words. Thus, if the network is able to retrieve images featuring instances of the target word, it shows that implicit segmentation was carried out at training time.

Results are shown in Figure 1. 40 words out of the 80 target words have a P@10$\geq$ 0.8. This shows that the network is able to map isolated words to their visual referent despite never having seen them in isolation and that the network implicitly segmented its input into sub-units.

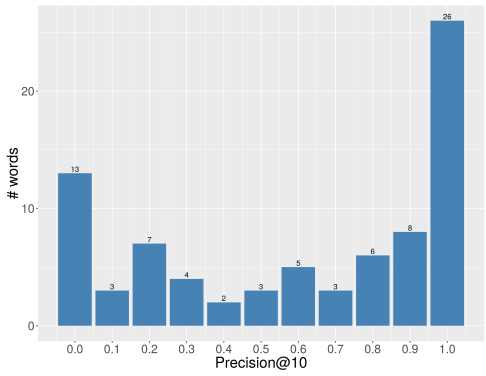


Figure 1: Precision@10 for the 80 isolated words corresponding to MSCOCO categories.

4.2 Factors Influencing Word Mapping

We explore here the factors that could come at play in the recognition of isolated words. We explore 2 types of factors: speech related factors and image related factors. For the former we consider

[4]List available at https://github.com/amikelive/coco-labels/blob/master/coco-labels-2014_2017.txt

[5]Evaluation is performed on the test set containing 5000 images

		Spearman's ρ	p-value	Effect
Images	Avg. Neighbour	-0.3906	0.0003 ***	Weak
	Avg. Size	0.3154	0.0043 **	Weak
	Avg. Freq	0.1187	0.4675	None
Text	Word Freq.	0.5514	1.148e-07 ***	Moderate
	# Syllables	-0.1211	0.2844	None

Table 2: Factors influencing word recognition performance in our model. Spearman's ρ between Precision@10 and mentioned variables as well as p-value.

word frequency (Word Freq.) and word length (# syllables). Concerning image related factors we consider object instances frequency in the images (Avg. Freq.), average number of neighbouring object instances (Avg. Neighbour), average area of each object (Avg. Size). Results are shown in Table 2. We observe a weak negative correlation between precison and average number of neighbouring objects, thus suggesting that objects that have a low number of neighbouring objects are better recognised by the network. It also seems that bigger objects yield better precison than smaller objects as we observe a weak positive correlation. Word frequency seems to play an important role as we observe a moderate positive correlation. However, we observe no correlation between precison and the length of the target words nor with object frequency in the images. Correlation values, however, remain relatively low, suggesting some other factors could also influence word recognition.

5 Word Activation

In this section we describe how individual words are activated by the network. To do so, we perform an ablation experiment (similar to that of Grosjean (1980) which was conducted on humans) where the neural model is inputted only with a truncated version of the 80 target words (see Section 5.1). Such a method is also called *gating* in the literature.

5.1 Gating

The gating paradigm *"involves the repeated presentation of a spoken stimulus (in this case, a word) such that its duration from onset is increased with each successive presentation"* (Cotton and Grosjean, 1984). In our case, it means the neural model is fed with truncated version of a target word, each truncated version comprising a larger part of the target word. Truncation is either done left-to-right (model only has access to the end of the word) or right-to-left (model only has access to the beginning of the word). Truncation is operated on the MFCC vectors computed for each individual word, meaning that MFCC vectors are iteratively removed either from the beginning of the word or the end of the word, but not from both sides at the same time. Each truncated version of the word is then fed to the speech encoder which outputs an embedding vector. As in our previous experiment, model is evaluated in terms of P@10.

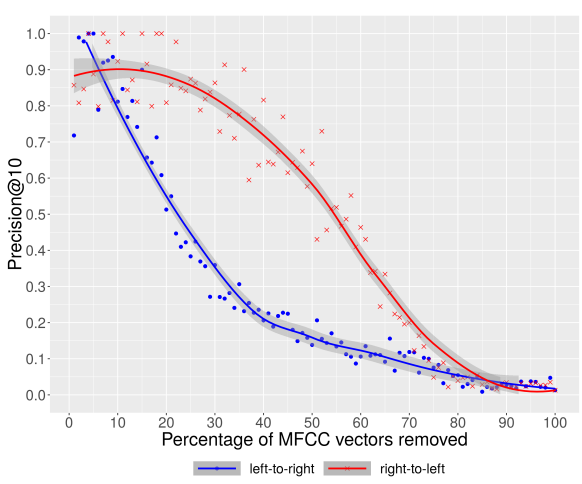
COHORT model, in its initial version (Marslen-Wilson, 1987), stipulates that word onset plays a crucial role in word recognition whereas other models of spoken word recognition give less importance to word onset. This importance of exact word onset matching was later revised in later COHORT models. The aim of this experiment is to test whether word onset plays a role in word recognition for the network or not. If it is the case, we expect the network to fail recovering images of the target word if the word is truncated left-to-right.

Figure 2a shows evolution of P@10 averaged over the 80 test words. As can be seen from the graph, precision evolves differently according to which part of the word was truncated. When the target words are truncated left-to-right, precision drops quickly. However, when truncated right-to-left, precision remains high before gradually dropping. These results show that the model is robust to truncation when it is carried out right-to-left but not when it is carried out left-to-right. Figure 2b shows the evolution of P@10 for one of the target words ("giraffe"). When MFCC vectors corresponding to the first phoneme are removed (/ʤ/), precision plummets from 1 to 0. However, when MFCC vectors belonging to the end of the word are removed, precision plateaus at 1 until /ɜˑ/ is reached and then plunges to 0. This shows the model successfully retrieved giraffe images when only prompted with /ʤɜˑ/ but not when prompted with /ɜˑæf/ even though the latter comprises a longer part of the target word.

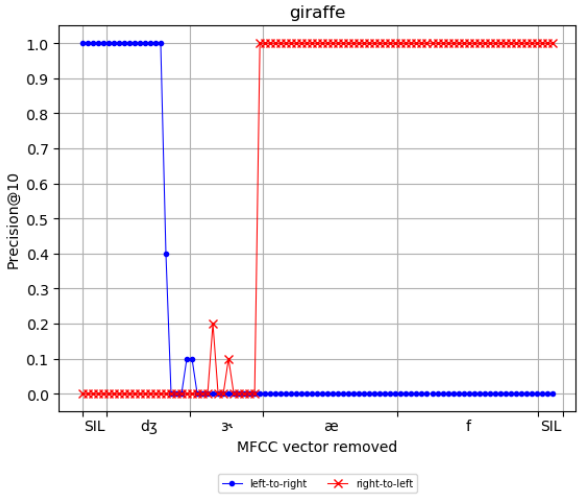
These results suggest that the model does not rely on a vague acoustic pattern to activate the semantic representation of a given concept, but needs to have access to the first phoneme in order to yield an appropriate representation.

5.2 Activated Pseudo-Words

Such ablation experiments also enables us to infer on what units the network relies to make its predictions. Indeed, Figure 3 allows us to see what



(a)



(b)

Figure 2: 2a Evolution of Precision@10 averaged over 80 test words as a function of the percentage of MFCC vector removed for each word. 2b Evolution of Precision@10 for each ablation step of the word "giraffe", with time-aligned phonemic transcription /dʒɜɹæf/ at the bottom. "SIL" signals silences. For both 2a and 2b, blue line displays scores when ablation was carried out left-to-right, meaning that at any given part on the blue curve, model has only had access to the rightmost part of the word. (e.g. /ɜɹæf/ without initial /dʒ/). Red line displays scores when ablation was carried out right-to-left, meaning that at any given part on the red curve, model has only had access to the leftmost part of the word. (e.g. /dʒɜɹ/ without final /æf/).

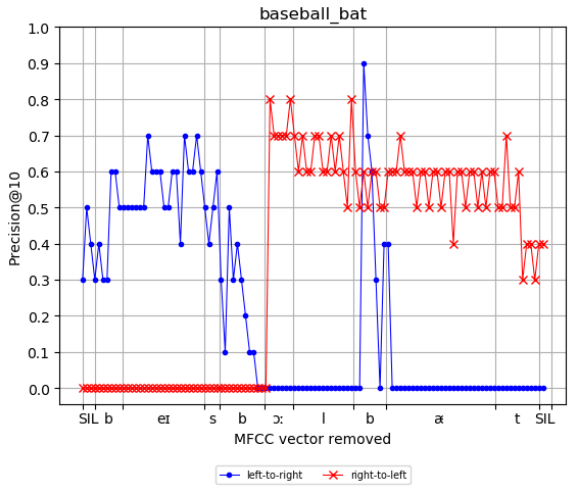


Figure 3: Evolution of P@10 for each ablation step of the word "baseball bat" with time aligned phonemic transcription /beizbɔːl#bæt/ at the bottom.

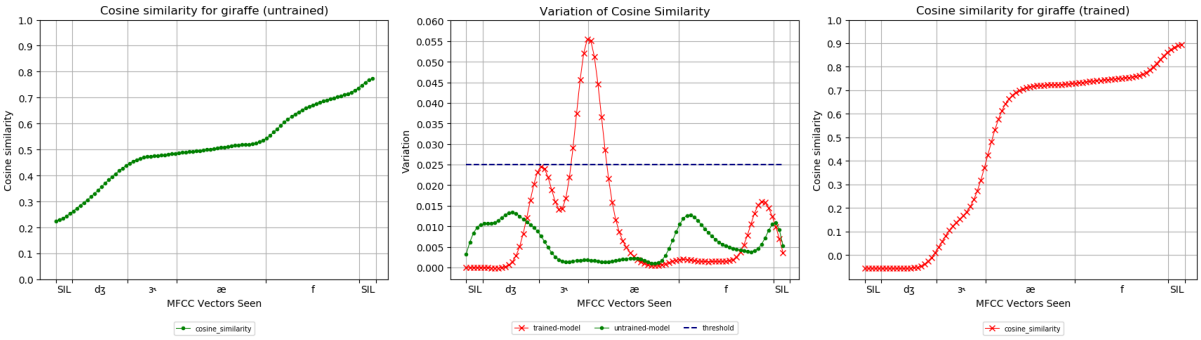
are the pseudo-words that were internalised by the network for the word "baseball bat". When truncation is done left-to-right (blue curve), we notice that at the beginning precision is quite high (≈ 0.6), then reaches 0 when only /ɔːlbæt/ is left, but suddenly increases up to 0.9 when the only part left is /bæt/. This suggests that the network mapped both "baseball bat" as a whole and "bat" as referring to the same object. We observed the same pattern for the word "fire hydrant" where both "fire hydrant" and "hydrant" are mapped to the same object.

However, Figure 2b shows that when only prompted with /dʒɜɹ/ the network manages to find pictures of giraffes. This suggests that the pseudo-words internalised by the network could be /dʒɜɹæf/ as a whole but might also be /dʒɜɹ/. We thus need to take caution when stating that the network has isolated words, as the words internalised by the network might not always match the human gold reference.

5.3 Gradual or Abrupt Activation?

Figure 2b shows that removing or adding one MFCC vector may yield large differences in the network performance. Precision decreases steeply and not steadily. This suggests that little acoustic differences yield wide differences in the final representation. Thus, in this section we analyse how representation is being constructed over time and explore if some MFCC vectors play a more important role than others in the activation of the final representation.

We progressively let the network see more and more of the MFCC vectors composing the word, iteratively feeding it with MFCC vectors starting from the beginning of the word until the network



(a) (b) (c)

Figure 4: Figure 4a shows evolution of the cosine similarity between the embeddings produced for each truncated version of the target word and the embedding for the full word using a model with randomly initialised weights. Figure 4c shows the same measure with the embeddings produced by a trained model. Figure 4b shows peaks indicating the inflection points of curve 4a (green) and 4c (red). For our experiments, we only considered inflection point to be significant if the resulting peak was higher than 0.025 (blue).

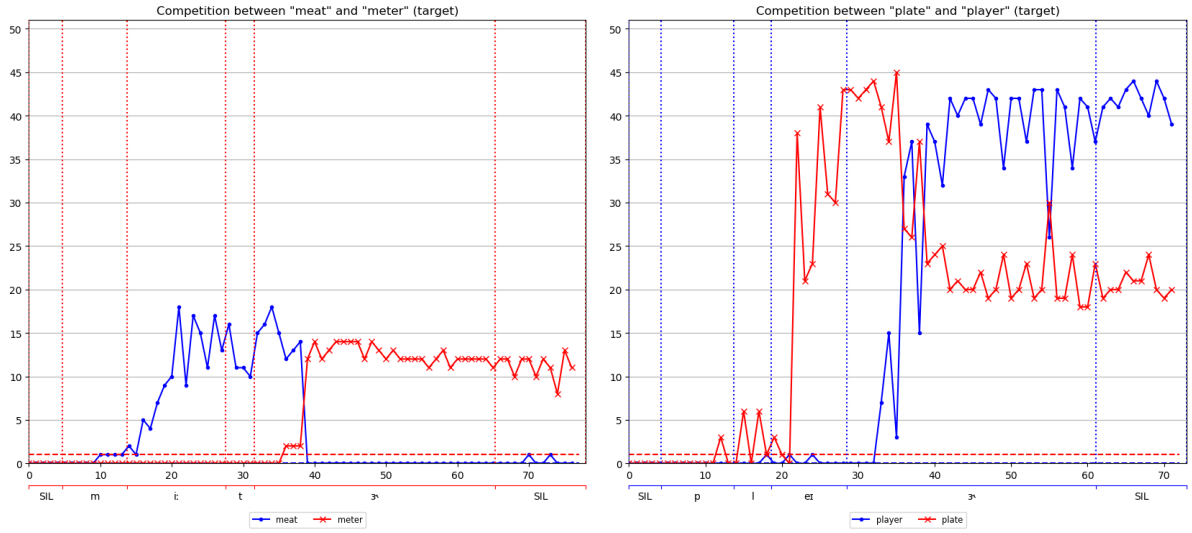
has had access to the full word. We then compute the cosine similarity between the embedding computed for each of the truncated version of the word and the embedding corresponding to the full word. The closer the cosine similarity is to 1, the more similar the two representations are. Thus, if each MFCC vector equally contributes to the final representation of the word, we expect cosine similarity to evolve linearly. However, if some MFCC vectors have a determining factor in the final representation we expect cosine similarity to evolve in steps rather than linearly. To detect steps that could occur in the evolution of cosine similarity, we approximate its derivative by computing first order difference. High steps should thus translate into peaks (e.g. Figure 4b). We compute the evolution of cosine similarity for the 80 target words encoded with the best trained model (e.g. Figure 4c) and also consider a baseline evolution by encoding the 80 target words with an untrained model (e.g. Figure 4a).[6] To avoid micro-steps of yielding peaks and thus creating noise, we smooth cosine evolution curves with a gaussian filter. We consider peaks higher than 0.025 as translating a high step in the evolution of cosine similarity.

On average, they are 1.35 peaks per word for the trained model against 0.1 peak per word for our baseline condition (untrained model), showing that cosine evolution is linear in the latter but not in the former. Thus, in our baseline condition (untrained model), each MFCC vector equally contributes to the final representation, whereas in our trained model some MFCC vectors are more decisive for the final representation than others. Indeed, some MFCC vectors trigger a high step in the cosine evolution suggesting that the embedding suddenly gets closer to its final value. Figure 4c shows the evolution of the cosine similarity for the word "giraffe". As it can be seen, cosine similarity does not tend linearly towards 1, but rather evolves in steps. Adding the MFCC vectors corresponding to the transition from /ɜɹ/ to /æ/ triggers a large difference in the embedding as the cosine similarity suddenly jumps to a higher value, showing it is getting closer to its final representation. However, cosine similarity plateaus once /æ/ is reached up until final silence, suggesting the final /æf/ plays little to no role in the final representation of the word.

6 Word Competition

As presented in Section 2.1, some linguistic models assume that the first phoneme of target word activates all the words starting by the same phoneme. The words that are activated but which do not correspond to the target words are called "competitors". As the listener perceives more and more of the target word some competitors are deactivated as they do not match what is being perceived. For example, considering the following lexicon: /beɪbi/ (baby), /beɪzɪk/ (basic), and /beɪzbɔːl/ (baseball), the first sound /b/ would activate all three words, once /beɪz/ is reached, "baby" would not be considered a competitor anymore, and once /beɪzb/ is reached the only word activated would be "baseball" as it is the only word whose beginning corresponds the perceived sounds.

[6] Thus consisting only of randomly initialised weights



(a) (b)

Figure 5: Illustration of lexical competition between 5a "meat" and "meter" (target) and 5b "plate" and "player" (target). Numbers in 1^{st} x-axis corresponds to the number of MFCC frames; 2^{nd} x-axis corresponds to time-aligned phonemic transcription of the target word; y-axis shows number of images for which at least one caption (out of 5) contains the target or competitor word. Vertical colour bars are projection of phoneme boundaries of the target word. Horizontal colour bars show chance score for each word (<2).

We test if the network displays such lexical competition patterns. To do so, we select a set of 29 word pairs according to the following criteria: i) words should at least appear 400 times or more in the captions of the training set, so that the network would have been able to learn a mapping between this word and its referent; ii) words forming a pair should at least start with the same phoneme;[7] iii) words should not be synonyms and clearly refer to a different visual object (thus excluding pairs such as "motorcycle" and "motorbike").

For each word pair, we select the longest word as target and progressively let the network see more and more of the MFCC vectors composing this word (as in Section 5.3). At each time step the network produces an embedding, which we use to rank the images from the closest matching image to the least matching image.[8] Then, for the 50 closest matching images, we check if at least one of the caption contains either the target word or the competitor. As the competitor is embedded in the longer word, we expect the network to produce an embedding close to that of the competitor at the beginning and then when the acoustic signal does not match the competitor anymore, we expect the network to be able to find only the target word.

Figure 5 shows example of competition between two word pairs. Figure 5a shows that when prompted with the beginning of the word "meter" /miːt/ the representation activated by the network is close to that of "meat" as the closest maching images's captions contain the word "meat". Representation of the word "meter" seems to be activated only when /ɜː/ is reached, and consequently triggers the total deactivation of the word "meat". Figure 5b displays a different pattern. As in the previous example, the beginning of the word "player" /pleɪ/ triggers the activation of the word "plate". When /ɜː/ is reached, the target word becomes activated and competitor "plate" starts to deactivate. However, the deactivation is not full, so that when the whole word "player" is entirely processed by the network, the word "plate" still remains highly activated. (REVISED) COHORT (Marslen-Wilson and Welsh, 1978; Marslen-Wilson, 1987) and TRACE (McClelland and Elman, 1986) both state that competing words are all activated at the same time, that is when the first phoneme is perceived. However here, the two competing words are activated sequentially but not at the same time. Also, in some cases, competing words that do not match the input anymore still remain highly activated.

[7] Phonemic transcription found in CMU Pronouncing Dictionnary was used

[8] That is, we compute the cosine distance between the embedding produced at time step t and all the images (5000) of our collection.

7 Conclusion

In this paper, we analysed the behaviour of a model of VGS and showed that a RNN-based model of VGS is able to map isolated words to their visual referents. This result is in line with previous results, such as that of Harwath and Glass (2019) which uses a CNN-based network. This shows that such models perform an implicit segmentation of the spoken input in order to extract the target words. However, the mechanism by which implicit segmentation is carried out and what cues are being used is still to be explained. We also demonstrated that not all words are equally well recognised and showed that word frequency and number of neighbouring object in an image partly explain this phenomenon.

Also, we introduced a methodology originating from linguistics to analyse the representation learned by neural networks: the gating paradigm. This enabled us to show that the beginning of a word can activate the representation of a given concept (e.g. /ʤɜ/ for "giraffe"). We explain this by the fact that the network has to handle a very small lexicon, where word forms rarely overlap and thus the network needs not see the full word to make its decision. More importantly, we showed that the network needs to have access to the first phoneme in order to activate the representation of the target word, thus showing that it does not respond to a vague acoustic pattern. Word onsets thus play a crucial role in the process of word activation and recognition for our network. Though word onsets are also important for humans, they are not as crucial as for our network. Indeed, humans are able to recover the missing information. In future work, we would like to test if sentential context has an effect in word recognition. We also demonstrated that our model is able to map multiple pseudo-words to the same referent such as humans do (Section 5.2). However, it is not clear how and when acoustics interface with meaning and this still remains an open question.

Finally, we showed that there is a form of lexical competition in the network. Indeed, small words embedded in longer words are activated. However, we showed that, contrary to humans where words sharing the same beginning are all activated at the same time, words are activated sequentially by the network. Also, some stay partially activated eventhough the input does not match that of the activated word.

Ultimately, we would like to highlight the fact that the gating paradigm could also by applied to understand the temporal dynamics of the representations learned by other speech architectures such as those used in speech recognition for instance.

Acknowledgments

This work was supported by grants from NeuroCoG IDEX UGA as part of the "Investissements d'avenir" program (ANR-15-IDEX-02).

References

Afra Alishahi, Marie Barking, and Grzegorz Chrupała. 2017. Encoding of phonology in a recurrent neural model of grounded speech. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 368–378. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nicholas Carlini and David A. Wagner. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. *2018 IEEE Security and Privacy Workshops (SPW)*, pages 1–7.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Grzegorz Chrupała, Lieke Gelderloos, and Afra Alishahi. 2017. Representations of language in a model of visually grounded speech signal. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 613–622. Association for Computational Linguistics.

Grzegorz Chrupała, Lieke Gelderloos, and Afra Alishahi. 2017. Synthetically spoken coco. [Data set] http://doi.org/10.5281/zenodo.400926.

Suzanne Cotton and François Grosjean. 1984. The gating paradigm: A comparison of successive and individual presentation formats. *Perception & Psychophysics*, 35(1):41–48.

Delphine Dahan and James S. Magnuson. 2006. Chapter 8 - spoken word recognition. In Matthew J.

Traxler and Morton A. Gernsbacher, editors, *Handbook of Psycholinguistics (Second Edition)*, second edition edition, pages 249 – 283. Academic Press, London.

Emmanuel Dupoux. 2018. Cognitive science in the era of artificial intelligence: A roadmap for reverse-engineering the infant language-learner. *Cognition*, 173:43 – 59.

François Grosjean. 1980. Spoken word recognition processes and the gating paradigm. *Perception & Psychophysics*, 28(4):267–283.

David Harwath and James Glass. 2019. Towards visually grounded sub-word speech unit discovery. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3017–3021.

David Harwath, Adrià Recasens, Dídac Surís, Galen Chuang, Antonio Torralba, and James Glass. 2018. Jointly discovering visual objects and spoken words from raw sensory input. In *Computer Vision – ECCV 2018*, pages 659–677, Cham. Springer International Publishing.

David F. Harwath, Antonio Torralba, and James R. Glass. 2016. Unsupervised learning of spoken language with visual context. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1858–1866.

William N. Havard, Jean-Pierre Chevrot, and Laurent Besacier. 2019. Models of visually grounded speech signal pay attention to nouns: A bilingual experiment on english and japanese. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8618–8622.

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Comput. Linguist.*, 43(4):761–780.

H. Kamper, G. Shakhnarovich, and K. Livescu. 2019. Semantic speech retrieval with a visually grounded model of untranscribed speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(1):89–98.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C. Lawrence Zitnick, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. 2014. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

William D. Marslen-Wilson. 1987. Functional parallelism in spoken word-recognition. *Cognition*, 25(1):71 – 102. Special Issue Spoken Word Recognition.

William D Marslen-Wilson and Alan Welsh. 1978. Processing interactions and lexical access during word recognition in continuous speech. *Cognitive Psychology*, 10(1):29 – 63.

James L McClelland and Jeffrey L Elman. 1986. The trace model of speech perception. *Cognitive Psychology*, 18(1):1 – 86.

Danny Merkx, Stefan L. Frank, and Mirjam Ernestus. 2019. Language Learning Using Speech to Image Retrieval. In *Proc. Interspeech 2019*, pages 1841–1845.

Dennis Norris. 1994. Shortlist: a connectionist model of continuous speech recognition. *Cognition*, 52:189–234.

Deb K. Roy and Alex P. Pentland. 2002. Learning words from sights and sounds: a computational model. *Cognitive Science*, 26(1):113–146.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, pages 1–1.

Andrea Weber and Odette Scharenborg. 2012. Models of spoken-word recognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 3(3):387–401.

Zhizheng Wu, Nicholas Evans, Tomi Kinnunen, Junichi Yamagishi, Federico Alegre, and Haizhou Li. 2014. Spoofing and countermeasures for speaker verification: a survey. *Elsevier Speech Communications, 2014*.

Wei Emma Zhang, Quan Z. Sheng, and Ahoud Abdulrahmn F. Alhazmi. 2019. Generating textual adversarial examples for deep learning models: A survey. *CoRR*, abs/1901.06796.

EQUATE : A Benchmark Evaluation Framework for Quantitative Reasoning in Natural Language Inference

Abhilasha Ravichander*, **Aakanksha Naik***,
Carolyn Rose, **Eduard Hovy**
Language Technologies Institute, Carnegie Mellon University
{aravicha, anaik, cprose, hovy}@cs.cmu.edu

Abstract

Quantitative reasoning is a higher-order reasoning skill that any intelligent natural language understanding system can reasonably be expected to handle. We present EQUATE[1] (Evaluating Quantitative Understanding Aptitude in Textual Entailment), a new framework for quantitative reasoning in textual entailment. We benchmark the performance of 9 published NLI models on EQUATE, and find that on average, state-of-the-art methods do not achieve an absolute improvement over a majority-class baseline, suggesting that they do not implicitly learn to reason with quantities. We establish a new baseline Q-REAS that manipulates quantities symbolically. In comparison to the best performing NLI model, it achieves success on numerical reasoning tests (+24.2%), but has limited verbal reasoning capabilities (-8.1%). We hope our evaluation framework will support the development of models of quantitative reasoning in language understanding.

1 Introduction

Numbers play a vital role in our lives. We reason with numbers in day-to-day tasks ranging from handling currency to reading news articles to understanding sports results, elections and stock markets. As numbers are used to communicate information accurately, reasoning with them is an essential core competence in understanding natural language (Levinson, 2001; Frank et al., 2008; Dehaene, 2011). A benchmark task in natural language understanding is natural language inference (NLI)(or recognizing textual entailment (RTE)) (Cooper et al., 1996; Condoravdi et al., 2003; Bos and Markert, 2005; Dagan et al., 2006), wherein a model determines if a natural language hypothesis can be justifiably inferred from a given premise[2]. Making such inferences often necessitates reasoning about quantities.

RTE-QUANT
P: After the deal closes, Teva will generate sales of about $ 7 billion a year, the company said.
H: Teva earns $ 7 billion a year.
AWP-NLI
P: Each of farmer Cunningham's 6048 lambs is either black or white and there are 193 white ones.
H: 5855 of Farmer Cunningham's lambs are black.
NEWSNLI
P: Emmanuel Miller, 16, and Zachary Watson, 17, are charged as adults, police said.
H: Two teen suspects charged as adults.
REDDITNLI
P: Oxfam says richest one percent to own more than rest by 2016.
H: Richest 1% To Own More Than Half Worlds Wealth By 2016 Oxfam.

Table 1: Examples from evaluation sets in EQUATE

Consider the following example from Table 1,

P: With 99.6% of precincts counted , Dewhurst held 48% of the vote to 30% for Cruz .
H: Lt. Gov. David Dewhurst fails to get 50% of primary vote.

To conclude the hypothesis is inferable, a model must reason that since 99.6% precincts are counted, even if all remaining precincts vote for Dewhurst, he would fail to get 50% of the primary vote. Scant attention has been paid to building datasets to evaluate this reasoning ability. To address this gap, we present EQUATE (Evaluating Quantity Understanding Aptitude in Textual Entailment) (§3), which consists of five evaluation sets, each

*The first two authors contributed equally to this work.
[1]Code and data available at https://github.com/AbhilashaRavichander/EQUATE.
[2]Often, this is posed as a three-way decision where the hypothesis can be inferred to be true (entailment), false (contradiction) or cannot be determined.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 349–361
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

featuring different facets of quantitative reasoning in textual entailment (Table 1) (including verbal reasoning with quantities, basic arithmetic computation, dealing with approximations and range comparisons.).

We evaluate the ability of existing state-of-the-art NLI models to perform quantitative reasoning (§4.1), by benchmarking 9 published models on EQUATE. Our results show that most models are incapable of quantitative reasoning, instead relying on lexical cues for prediction. Additionally, we build Q-REAS , a shallow semantic reasoning baseline for quantitative reasoning in NLI (§4.2). Q-REAS is effective on synthetic test sets which contain more quantity-based inference, but shows limited success on natural test sets which require deeper linguistic reasoning. However, the hardest cases require a complex interplay between linguistic and numerical reasoning. The EQUATE evaluation framework makes it clear where this new challenge area for textual entailment stands.

2 Related Work

NLI has attracted community-wide interest as a stringent test for natural language understanding (Cooper et al., 1996; Fyodorov; Glickman et al., 2005; Haghighi et al., 2005; Harabagiu and Hickl, 2006; Romano et al., 2006; Dagan et al., 2006; Giampiccolo et al., 2007; Zanzotto et al., 2006; Malakasiotis and Androutsopoulos, 2007; MacCartney, 2009; de2; Dagan et al., 2010; Angeli and Manning, 2014; Marelli et al., 2014). Recently, the creation of large-scale datasets (Bowman et al., 2015; wil; Khot et al., 2018) spurred the development of many neural models (Parikh et al., 2016; Nie and Bansal, 2017; Conneau et al., 2017; Balazs et al., 2017; Chen et al., 2017a; Radford et al., 2018; Devlin et al., 2018).

However, state-of-the-art models for NLI treat the task like a matching problem, which appears to work in many cases, but breaks down in others. As the field moves past current models of the matching variety to ones that embody more of the reasoning we know is part of the task, we need benchmarks that will enable us to mark progress in the field. Prior work on challenge tasks has already made headway in defining tasks for subproblems such as lexical inference with hypernymy, co-hyponymy, antonymy (Glockner et al., 2018; Naik et al., 2018). In this work, we specifically probe into quantitative reasoning.

De Marneffe et al. (2008) find that in a corpus of real-life contradiction pairs collected from Wikipedia and Google News, *29%* contradictions arise from numeric discrepancies, and in the RTE-3 (Recognizing Textual Entailment) development set, numeric contradictions make up 8.8% of contradictory pairs. Naik et al. (2018) find that model inability to do numerical reasoning causes 4% of errors made by state-of-the-art models. Sammons et al. (2010); Clark (2018) argue for a systematic knowledge-oriented approach in NLI by evaluating specific semantic analysis tasks, identifying quantitative reasoning in particular as a focus area. Bentivogli et al. (2010) propose creating specialized datasets, but feature only 6 examples with quantitative reasoning. Our work bridges this gap by providing a more comprehensive examination of quantitative reasoning in NLI.

While to the best of our knowledge, prior work has not studied quantitative reasoning in NLI, Roy (2017) propose a model for a related subtask called *quantity entailment*, which aims to determine if a given quantity can be inferred from a sentence. In contrast, our work is concerned with general-purpose textual entailment which considers if a given *sentence* can be inferred from another. Our work also relates to solving arithmetic word problems (Hosseini et al., 2014; Mitra and Baral, 2016; Zhou et al., 2015; Upadhyay et al., 2016; Huang et al., 2017; Kushman et al., 2014a; Koncel-Kedziorski et al., 2015; roy; Roy, 2017; Ling et al., 2017a). A key difference is that word problems focus on arithmetic reasoning, while the requirement for linguistic reasoning and world knowledge is limited as the text is concise, straightforward, and self-contained (Hosseini et al., 2014; Kushman et al., 2014b). Our work provides a testbed that evaluates basic arithmetic reasoning while incorporating the complexity of natural language.

Recently, Dua et al. (2019) also recognize the importance of quantitative reasoning for text understanding. They propose DROP, a reading comprehension dataset focused on a limited set of discrete operations such as counting, comparison, sorting and arithmetic. In contrast, EQUATE features diverse phenomena that occur naturally in text, including reasoning with approximation, ordinals, implicit quantities and quantifiers, requiring NLI models to reason comprehensively about the interplay between quantities and language. Ad-

ditionally, through EQUATE we suggest the inclusion of controlled synthetic tests in evaluation benchmarks. Controlled tests act as basic validation of model behaviour, isolating model ability to reason about a property of interest.

3 Quantitative Reasoning in NLI

Our interpretation of "quantitative reasoning" draws from cognitive testing and education (Stafford, 1972; Ekstrom et al., 1976), which considers it "verbal problem-solving ability". While inextricably linked to mathematics, it is an inclusive skill involving everyday language rather than a specialized lexicon. To excel at quantitative reasoning, one must interpret quantities expressed in language, perform basic calculations, judge their accuracy, and justify quantitative claims using verbal and numeric reasoning. These requirements show a reciprocity: NLI lends itself as a test bed for quantitative reasoning, which conversely, is important for NLI (Sammons et al., 2010; Clark, 2018). Motivated by this, we present the EQUATE (Evaluating Quantity Understanding Aptitude in Textual Entailment) framework.

3.1 The EQUATE Dataset

EQUATE consists of five NLI test sets featuring quantities. Three of these tests for quantitative reasoning feature language from real-world sources such as news articles and social media (§3.2; §3.3; §3.4). We focus on sentences containing quantities with numerical values, and consider an entailment pair to feature quantitative reasoning if it is at least one component of the reasoning required to determine the entailment label (but not necessarily the only reasoning component). Quantitative reasoning features quantity matching, quantity comparison, quantity conversion, arithmetic, qualitative processes, ordinality and quantifiers, quantity noun and adverb resolution[3] as well as verbal reasoning with the quantity's textual context[4]. Appendix B gives some examples for these quantitative phenomena. We further filter sentence pairs which require only temporal reasoning, since specialized knowledge is needed to reason about time. These three test sets contain pairs which conflate multiple lexical and quantitative reasoning phenomena. In order to study aspects of quantitative reasoning in isolation, EQUATE further features two controlled synthetic tests (§3.5; §3.6).

3.2 RTE-Quant

This test set is constructed from the RTE subcorpus for quantity entailment (Roy, 2017), originally drawn from the RTE2-RTE4 datasets (Dagan et al., 2006). The original sub-corpus conflates temporal and quantitative reasoning. We discarded pairs requiring temporal reasoning, obtaining a set of 166 entailment pairs.

3.3 NewsNLI

This test set is created from the CNN corpus (Hermann et al., 2015) of news articles with abstractive summaries. We identify summary points with quantities, filtering out temporal expressions. For a summary point, the two most similar sentences[5] from the article are chosen, flipping pairs where the premise begins with a first-person pronoun (eg:⟨"He had nine pears", "Bob had nine pears"⟩ becomes ⟨"Bob had nine pears", "He had nine pears"⟩). The top 50% of similar pairs are retained to avoid lexical overlap bias. We crowdsource annotations for a subset of this data from Amazon Mechanical Turk. Crowdworkers[6] are shown two sentences and asked to determine whether the second sentence is definitely true, definitely false, or not inferable given the first. We collect 5 annotations per pair, and consider pairs with lowest token overlap between premise and hypothesis and least difference in premise-hypothesis lengths when stratified by entailment label. Top 1000 samples meeting these criteria form our final set. To validate crowdsourced labels, experts are asked to annotate 100 pairs. Crowdsourced gold labels match expert gold labels in 85% cases, while individual crowdworker labels match expert gold labels in 75.8%. Disagreements are manually resolved by experts and examples not featuring quantitiative reasoning are filtered, leaving a set of 968 samples.

3.4 RedditNLI

This test set is sourced from the popular social forum \reddit[7]. Since reasoning about quanti-

[3] Such as the quantities represented in *dozen*, *twice*, *teenagers*.

[4] For example, ⟨Obama cuts tax rate to 28%, Obama wants to cut tax rate to 28% as part of overhaul⟩.

[5] According to Jaccard similarity.

[6] We require crowdworkers to have an approval rate of 95% on at least 100 tasks and pass a qualification test.

[7] According to the Reddit User Agreement, users grant Reddit the right to make their content available to other organizations or individuals.

Source	Test Set	Size	Classes	Data Source	Annotation Source	Quantitative Phenomena
Natural	RTE-Quant	166	2	RTE2-RTE4	Experts	Arithmetic, Ranges, Quantifiers
	NewsNLI	968	2	CNN	Crowdworkers	Ordinals, Quantifiers, Arithmetic, Approximation, Magnitude, Ratios
	RedditNLI	250	3	Reddit	Experts	Range, Arithmetic, Approximation, Verbal
Synthetic	Stress Test	7500	3	AQuA-RAT	Automatic	Quantifiers
	AwpNLI	722	2	Arithmetic Word Problems	Automatic	Arithmetic

Table 2: An overview of test sets included in EQUATE. RedditNLI and Stress Test are framed as 3-class (entailment, neutral, contradiction) while RTE-Quant, NewsNLI and AwpNLI are 2-class (entails=yes/no). RTE 2-4 formulate entailment as a 2-way decision. We find that few news article headlines are contradictory, thus NewsNLI is similarly framed as a 2-way decision. For algebra word problems, substituting the wrong answer in the hypothesis necessarily creates a contradiction under the event coreference assumption (De Marneffe et al., 2008), thus it is framed as a 2-way decision as well.

ties is important in domains like finance or economics, we scrape all headlines from the posts on \r\economics, considering titles that contain quantities and do not have meta-forum information. Titles appearing within three days of each other are clustered by Jaccard similarity, and the top 300 pairs are extracted. After filtering out nonsensical titles, such as concatenated stock prices, we are left with 250 sentence pairs. Similar to RTE, two expert annotators label these pairs, achieving a Cohen's kappa of 0.82. Disagreements are discussed to resolve final labels.

3.5 Stress Test

We include the numerical reasoning stress test from (Naik et al., 2018) as a synthetic sanity check. The stress test consists of 7500 entailment pairs constructed from sentences in algebra word problems (Ling et al., 2017b). Focusing on quantifiers, it requires models to compare entities from hypothesis to the premise while incorporating quantifiers, but does not require them to perform the computation from the original algebra word problem (eg: ⟨"NHAI employs 100 men to build a highway of 2 km in 50 days working 8 hours a day","NHAI employs less than 700 men to build a highway of 2 km in 50 days working 8 hours a day"⟩).

3.6 AwpNLI

To evaluate arithmetic ability of NLI models, we repurpose data from arithmetic word problems (roy). They have the following characteristic structure. First, they establish a world and optionally update its state. Then, a question is posed about the world. This structure forms the basis of our pair creation procedure. World building and update statements form the premise. A hypothesis template is generated by identifying modal/auxiliary verbs in the question, and subsequent verbs, which we call secondary verbs. We identify the agent and conjugate the secondary verb in present tense followed by the identified unit to form the final template (for example, the algebra word problem 'Gary had 73.0 dollars. He spent 55.0 dollars on a pet snake. How many dollars did Gary have left?' would generate the hypothesis template 'Agent(Gary) Verb(Has) Answer(18.0) Unit(dollars) left'). For every template, the correct guess is used to create an entailed hypothesis. Contradictory hypotheses are created by randomly sampling a wrong guess ($x \in \mathbb{Z}^+$ if correct guess is an integer, and $x \in \mathbb{R}^+$ if it is a real number) [8]. We check for grammaticality, finding only 2% ungrammatical hypotheses, which are manually corrected leaving a set of 722 pairs.

4 Models

We describe the 9 NLI models[9] used in this study, as well as our new baseline. The interested reader is invited to refer to the corresponding publications for further details.

4.1 NLI Models

1) **Majority Class (MAJ):** Simple baseline that always predicts the majority class in test set.

[8] From a uniform distribution over an interval of 10 around the correct guess (or 5 for numbers less than 5), to identify plausible wrong guesses.

[9] Accuracy of all models on MultiNLI closely matches original publications (numbers in appendix A).

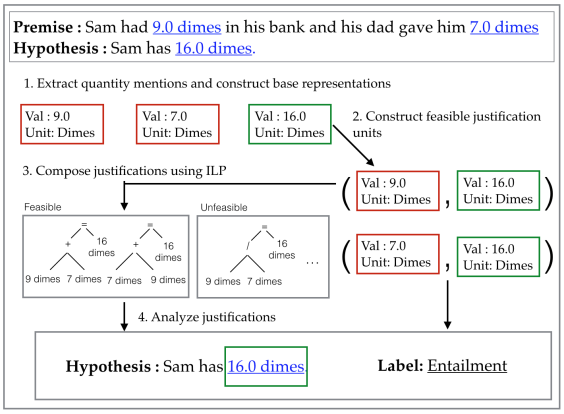


Figure 1: Overview of Q-REAS baseline.

2) **Hypothesis-Only (HYP):** FastText classifier (Mikolov et al., 2018) trained on only hypotheses (Gururangan et al., 2018).
3) **ALIGN:** A bag-of-words alignment model inspired by MacCartney (2009).[10]
4) **CBOW:** A simple bag-of-embeddings sentence representation model (wil).
5) **BiLSTM:** The simple BiLSTM model described by wil.
6) **Chen (CH):** Stacked BiLSTM-RNNs with shortcut connections and character word embeddings (Chen et al., 2017b).
7) **InferSent:** A single-layer BiLSTM-RNN model with max-pooling (Conneau et al., 2017).
8) **SSEN:** Stacked BiLSTM-RNNs with shortcut connections (Nie and Bansal, 2017).
9) **ESIM:** Sequential inference model proposed by Chen et al. (2017a) which uses BiLSTMs with an attention mechanism.
10) **OpenAI GPT:** Transformer-based language model (Vaswani et al., 2017), with finetuning on NLI (Radford et al., 2018).
11) **BERT:** Transformer-based language model (Vaswani et al., 2017), with a cloze-style and next-sentence prediction objective, and finetuning on NLI (Devlin et al., 2018).

4.2 Q-REAS Baseline System

Figure 1 describes the Q-REAS baseline for quantitative reasoning in NLI. The model manipulates quantity representations symbolically to make entailment decisions, and is intended to serve as a strong heuristic baseline for numerical reasoning on the EQUATE benchmark. This model has four stages: Quantity mentions are extracted and parsed into semantic representations called NUMSETS (§4.2.1, §4.2.2); compatible NUMSETS are extracted (§4.2.3) and composed (§4.2.4) to form *justifications*; Justifications are analyzed to determine entailment labels (§4.2.5).

[10]Model accuracy on RTE-3 test is 61.12%, comparable to the reported average model performance in the RTE competition of 62.4% .

INPUT	
P_c	Set of "compatible" single-valued premise quantities
P_r	Set of "compatible" range-valued premise quantities
H	Hypothesis quantity
O	Operator set $\{+, -, *, /, =, \cap, \cup, \setminus, \subseteq\}$
L	Length of equation to be generated
SL	Symbol list ($P_c \cup P_r \cup H \cup O$)
TL	Type list (set of types from P_c, P_r, H)
N	Length of symbol list
K	Index of first range quantity in symbol list
M	Index of first operator in symbol list
OUTPUT	
e_i	Index of symbol assigned to i^{th} position in postfix equation
VARIABLES	
x_i	Main ILP variable for position i
c_i	Indicator variable: is e_i a single value?
r_i	Indicator variable: is e_i a range?
o_i	Indicator variable: is e_i an operator?
d_i	Stack depth of e_i
t_i	Type index for e_i

Table 3: Input, output and variable definitions for the Integer Linear Programming (ILP) framework used for quantity composition

4.2.1 Quantity Segmenter

We follow Barwise and Cooper (1981) in defining quantities as having a number, unit, and an optional approximator. Quantity mentions are identified as least ancestor noun phrases from the constituency parse of the sentence containing cardinal numbers.

4.2.2 Quantity Parser

The quantity parser constructs a grounded representation for each quantity mention in the premise or hypothesis, henceforth known as a NUMSET [11]. A NUMSET is a tuple (val, unit, ent, adj, loc, verb, freq, flux)[12] with:

1. val $\in [\mathbb{R}, \mathbb{R}]$: quantity value represented as a range
2. unit $\in S$: unit noun associated with quantity
3. ent $\in S^{\phi}$: entity noun associated with unit (e.g., *donations* worth 100$)

[11]A NUMSET may be a composition of other NUMSETS .

[12]As in (Koncel-Kedziorski et al., 2015) S denotes all possible spans in the sentence, ϕ represents the empty span, and S^{ϕ}=$S \cup \phi$

Definitional Constraints	
Range restriction	$x_i < K$ or $x_i = M-1$ for $i \in [0, L-1]$ if $c_i = 1$
	$x_i \geq K$ and $x_i < M$ for $i \in [0, L-1]$ if $r_i = 1$
	$x_i \geq M$ for $i \in [0, L-1]$ if $o_i = 1$
Uniqueness	$c_i + r_i + o_i = 1$ for $i \in [0, L-1]$
Stack definition	$d_0 = 0$ (Stack depth initialization)
	$d_i = d_{i-1} - 2o_i + 1$ for $i \in [0, L-1]$ (Stack depth update)
Syntactic Constraints	
First two operands	$c_0 + r_0 = 1$ and $c_1 + r_1 = 1$
Last operator	$x_{L-1} \geq N-1$ (Last operator should be one of $\{=, \subseteq\}$)
Last operand	$x_{L-2} = M-1$ (Last operand should be hypothesis quantity)
Other operators	$x_i \leq N-2$ for $i \in [0, L-3]$ if $o_i = 1$
Other operands	$x_i < K$ for $i \in [0, L-3]$ if $c_i = 1$
	$x_i < M$ for $i \in [0, L-3]$ if $r_i = 1$
Empty stack	$d_{L-1} = 0$ (Non-empty stack indicates invalid postfix expression)
Premise usage	$x_i \neq x_j$ for $i, j \in [0, L-1]$ if $o_i \neq 1, o_j \neq 1$
Operand Access	
Right operand	$op2(x_i) = x_{i-1}$ for $i \in [0, L-1]$ such that $o_i = 1$
Left operand	$op1(x_i) = x_l$ for $i, l \in [0, L-1]$ where $o_i = 1$ and l is the largest index such that $l \leq (i-2)$ and $d_l = d_i$

Table 4: Mathematical validity constraint definitions for the ILP framework. Functions $op1()$ and $op2()$ return the left and right operands for an operator respectively. Variables defined in table 3.

Type Consistency Constraints	
Type assignment	$t_i = TL[k]$ for $i \in [0, L-1]$ if $c_i + r_i = 1$ and $type(SL_i) = k$
Two type match	$t_i = t_a = t_b$ for $i \in [0, L-1]$ such that $o_i = 1, x_i \in \{+, -, *, /, =, \cap, \cup, \setminus, \subseteq\}, a = op1(x_i), b = op2(x_i)$
One type match	$t_i \in \{t_a, t_b\}, t_a \neq t_b$ for $i \in [0, L-1]$ such that $o_i = 1, x_i = *, a = op1(x_i), b = op2(x_i)$
	$t_i = t_a \neq t_b$ for $i \in [0, L-1]$ such that $o_i = 1, x_i = /, a = op1(x_i), b = op2(x_i)$
Operator Consistency Constraints	
Arithmetic operators	$c_a = c_b = 1$ for $i \in [0, L-1]$ such that $o_i = 1, x_i \in \{+, -, *, /, =\}, a = op1(x_i), b = op2(x_i)$
Range operators	$r_a = r_b = 1$ for $i \in [0, L-1]$ such that $o_i = 1, x_i \in \{\cap, \cup, \setminus\}, a = op1(x_i), b = op2(x_i)$
	$r_b = 1$ for $i \in [0, L-1]$ such that $o_i = 1, x_i = \subseteq, b = op2(x_i)$

Table 5: Linguistic consistency constraint definitions for the ILP framework. Functions $op1()$ and $op2()$ return the left and right operands for an operator respectively. Variables defined in table 3.

4. adj $\in S^{\phi}$: adjective associated with unit if any[13],
5. loc $\subseteq S^{\phi}$: location of unit (e.g.,'in the bag')[14]
6. verb $\in S^{\phi}$: action verb associated with quantity[15].
7. freq $\subseteq S^{\phi}$: if quantity recurs[16] (e.g, 'per hour'),
8. flux $\in \{$increase to, increase from, decrease to, decrease from$\}^{\phi}$: if quantity is in a state of flux[17].

To extract **values** for a quantity, we extract cardinal numbers, recording contiguity. We normalize the number[18]. We also handle simple ratios such as quarter, half etc, and extract bounds (eg: *fewer than 10 apples* is parsed to $[-\infty, 10]$ apples.)

To extract **units**, we examine tokens adjacent to cardinal numbers in the quantity mention and identify known units. If no known units are found, we assign the token in a *numerical modifier* relationship with the cardinal number, else we assign the nearest noun to the cardinal number as the unit. A quantity is determined to be **approximate** if the word in an *adverbial modifier* relation with the cardinal number appears in a gazetteer[19]. If approximate, range is extended to (+/-)2% of the

[13]Extracted as governing verb linked to entity by an *amod* relation.

[14]Extracted as prepositional phrase attached to the quantity and containing noun phrase.

[15]Extracted as governing verb linked to entity by *dobj* or *nsubj* relation.

[16]extracted using keywords *per* and *every*

[17]using gazetteer: *increasing*, *rising*, *rose*, *decreasing*, *falling*, *fell*, *drop*

[18](remove ","s, convert written numbers to float, decide the numerical value, for example hundred fifty eight thousand is 158000, two fifty eight is 258, 374m is 3740000 etc.). If cardinal numbers are non-adjacent, we look for an explicitly mentioned range such as 'to' and 'between'.

[19]roughly, approximately, about, nearly, roundabout, around, circa, almost, approaching, pushing, more or less, in the neighborhood of, in the region of, on the order of,something like, give or take (a few), near to, close to, in the ballpark of

M \ D	RTE-Q	Δ	NewsNLI	Δ	RedditNLI	Δ	NR ST	Δ	AWPNLI	Δ	Nat. Avg. Δ	Synth. Avg. Δ	All Avg. Δ
MAJ	57.8	0.0	50.7	0.0	**58.4**	**0.0**	33.3	0.0	50.0	0.0	+0.0	+0.0	+0.0
HYP	49.4	-8.4	52.5	+1.8	40.8	-17.6	31.2	-2.1	50.1	+0.1	-8.1	-1.0	-5.2
ALIGN	62.1	+4.3	56.0	+5.3	34.8	-23.6	22.6	-10.7	47.2	-2.8	-4.7	-6.8	-5.5
CBOW	47.0	-10.8	61.8	+11.1	42.4	-16.0	30.2	-3.1	50.7	+0.7	-5.2	-1.2	-3.6
BiLSTM	51.2	-6.6	63.3	+12.6	50.8	-7.6	31.2	-2.1	50.7	+0.7	-0.5	-0.7	-0.6
CH	54.2	-3.6	64.0	+13.3	55.2	-3.2	30.3	-3.0	50.7	+0.7	+2.2	-1.2	+0.9
InferSent	66.3	+8.5	65.3	+14.6	29.6	-28.8	28.8	-4.5	50.7	+0.7	-1.9	-1.9	-1.9
SSEN	58.4	+0.6	65.1	+14.4	49.2	-9.2	28.4	-4.9	50.7	+0.7	+1.9	-2.1	+0.3
ESIM	54.8	-3.0	62.0	+11.3	45.6	-12.8	21.8	-11.5	50.1	+0.1	-1.5	-5.7	-3.2
GPT	**68.1**	**+10.3**	72.2	+21.5	52.4	-6.0	36.4	+3.1	50.0	+0.0	+8.6	+1.6	+5.8
BERT	57.2	-0.6	**72.8**	**+22.1**	49.6	-8.8	36.9	+3.6	42.2	-7.8	+4.2	-2.1	+1.7
Q-REAS	56.6	-1.2	61.1	+10.4	50.8	-7.6	**63.3**	**+30**	**71.5**	**+21.5**	+0.5	+25.8	**+10.6**

Table 6: Accuracies(%) of 9 NLI Models on five tests for quantitiative reasoning in entailment. M and D represent *models* and *datasets* respectively. Δ captures improvement over majority-class baseline for a dataset. Column Nat.Avg. reports the average accuracy(%) of each model across 3 evaluation sets constructed from natural sources (RTE-Quant, NewsNLI, RedditNLI), whereas Synth.Avg. reports the average accuracy(%) on 2 synthetic evaluation sets (Stress Test, AwpNLI). Column Avg. represents the average accuracy(%) of each model across all 5 evaluation sets in EQUATE.

current value.

4.2.3 Quantity Pruner

The pruner constructs "compatible" premise-hypothesis NUMSET pairs. Consider the pair "Insurgents killed *7 U.S. soldiers*, set off a car bomb that killed *four Iraqi policemen*" and "*7 US soldiers* were killed, and *at least 10 Iraqis* died". Our parser extracts NUMSETS corresponding to *"four Iraqi policemen"* and *"7 US soldiers"* from premise and hypothesis respectively. But these NUMSETS should not be compared as they involve different units. The pruner discards such incompatible pairs. Heuristics to identify unit-compatible NUMSET pairs include three cases- 1) direct string match, 2) synonymy/hypernymy relations from WordNet, 3) one unit is a nationality/ job[20] and the other unit is synonymous with person (Roy, 2017).

4.2.4 Quantity Composition

The composition module detects whether a hypothesis NUMSET is justified by composing "compatible" premise NUMSETS . For example, consider the pair "I had *3 apples* but gave *one* to my brother" and "I have *two apples*". Here, the premise NUMSETS P_1 ("*3 apples*") and P_2 ("*one apple*") must be composed to deduce that the hypothesis NUMSET H_1 ("*2 apples*") is justified. Our framework accomplishes this by generating postfix arithmetic equations[21] from premise NUMSETS , that justify the hypothesis NUMSET [22]. In this example, the expression $< P_1, P_2, -, H_1, =>$ will be generated.

The set of possible equations is exponential in number of NUMSETS , making exhaustive generation intractable. But a large number of equations are invalid as they violate constraints such as unit consistency. Thus, our framework uses integer linear programming (ILP) to constrain the equation space. It is inspired by prior work on algebra word problems (Koncel-Kedziorski et al., 2015), with some key differences:

1. **Arithmetic equations:** We focus on arithmetic equations instead of algebraic ones.
2. **Range arithmetic:** Quantitative reasoning involves ranges, which are handled by representing then as endpoint-inclusive intervals and adding the four operators ($\cup, \cap, \setminus, \subseteq$)
3. **Hypothesis quantity-driven:** We optimize an ILP model for each hypothesis NUMSET because a sentence pair is marked "entailment" iff every hypothesis quantity is justified.

Table 3 describes ILP variables. We impose the following types of constraints:
1. Definitional Constraints: Ensure that ILP variables take on valid values by constraining initialization, range, and update.
2. Syntactic Constraints: Assure syntactic validity of generated postfix expressions by limiting

[20]Lists of jobs, nationalities scraped from Wikipedia.

[21]Note that arithmetic equations differ from algebraic equations in that they do *not* contain unknown variables

[22]Direct comparisons are incorporated by adding "=" as an operator.

operator-operand ordering.

3. Operand Access: Simulate stack-based evaluation correctly by choosing correct operator-operand assignments.

4. Type Consistency: Ensure that all operations are type-compatible.

5. Operator Consistency: Force range operators to have range operands and mathematical operators to have single-valued operands.

Definitional, syntactic, and operand access constraints ensure mathematical validity while type and operator consistency constraints add linguistic consistency. Constraint formulations are provided in Tables 4 and 5. We limit tree depth to 3 and retrieve a maximum of 50 solutions per hypothesis NUMSET , then solve to determine whether the equation is mathematically correct. We discard equations that use invalid operations (division by 0) or add unnecessary complexity (multiplication/ division by 1). The remaining equations are considered plausible justifications .

4.2.5 Global Reasoner

The global reasoner predicts the final entailment label as shown in Algorithm 1[23], on the assumption that every NUMSET in the hypothesis *has* to be justified [24] for entailment.

Algorithm 1 PredictEntailmentLabel(P, H, C, E)

Input: Premise quantities P, Hypothesis quantities H, Compatible pairs C, Equations E
Output: Entailment label $l \in$ { e, c, n }

if $C = \emptyset$ **then return** n
$J \leftarrow \emptyset$
$L \leftarrow []$
for $q_h \in H$ **do**
    $J_h \leftarrow \{q_p \mid q_p \in P, (q_p, q_h) \in C\}$
    $J \leftarrow J \cup \{(q_h, J_h)\}$
    $L \leftarrow L + [false]$
for $(q_h, J_h) \in J$ **do**
    if $J_h = \emptyset$ **then return** n
    for $q_p \in J_h$ **do**
        $s \leftarrow$ MaxSimilarityClass(q_p, q_h)
        if $s = e$ **then**
            if ValueMatch(q_p, q_h) **then**
                $L[q_h] = true$
            if !ValueMatch(q_p, q_h) **then**
                $L[q_h] = false$
        if $s = c$ **then**
            if ValueMatch(q_p, q_h) **then**
                $L[q_h] = c$
for $q_h \in H$ **do**
    $E_q \leftarrow \{e_i \in E \mid hyp(e_i) = q_h\}$
    if $E_q \neq \emptyset$ **then**
        $L[q_h] = true$
if $c \in L$ **then return** c
if count($L, true$) = len(L) **then return** e
return n

5 Results and Discussion

Table 6 presents results on EQUATE. All models, except Q-REAS are trained on MultiNLI. Q-REAS utilizes WordNet and lists from Wikipedia. We observe that neural models, particularly OpenAI GPT excel at verbal aspects of quantitative reasoning (RTE-Quant, NewsNLI), whereas Q-REAS excels at numerical aspects (Stress Test, AwpNLI).

5.1 Neural Models on NewsNLI:

To tease apart contributory effects of numerical and verbal reasoning in natural data, we experiment with NewsNLI. We extract all entailed pairs where a quantity appears in both premise and hypothesis, and perturb the quantity in the hypothesis generating contradictory pairs. For example, the pair ⟨'In addition to 79 fatalities , some 170 passengers were injured.'⟩ 'The crash took the lives of 79 people and injured some 170', 'entailment' is changed to ⟨'In addition to 79 fatalities , some 170 passengers were injured.', 'The crash took the lives of 80 people and injured some 170', 'contradiction'⟩, assuming scalar implicature and event coreference. Our perturbed test set contains 218 pairs. On this set, GPT[25] achieves an accuracy of 51.18%, as compared to 72.04% on the unperturbed set, suggesting the model relies on verbal cues rather than numerical reasoning. In comparison, Q-REAS achieves an accuracy of 98.1% on the perturbed set, compared to 75.36% on the unperturbed set, highlighting

[23] MaxSimilarityClass() takes two quantities and returns a probability distribution over entailment labels based on unit match. Similarly, ValueMatch() detects whether two quantities match in value (this function can also handle ranges).

[24] This is a necessary but not sufficient condition for entailment. Consider the example, ⟨'Sam believed Joan had 5 apples', 'Joan had 5 apples'⟩. The hypothesis quantities of 5 apples is justified but is not a sufficient condition for entailment.

[25] the best-performing neural model on EQUATE.

reliance on quantities rather than verbal information. Closer examination reveals that OpenAI switches to predicting the 'neutral' category for perturbed samples instead of entailment, accounting for 42.7% of its errors, possibly symptomatic of lexical bias issues (Naik et al., 2018).

5.2 What Quantitative Phenomena Are Hard?

We sample 100 errors made by Q-REAS on each test in EQUATE, to identify phenomena not addressed by simple quantity comparison. Our analysis of causes for error suggest avenues for future research:

1. **Multi-step numerical-verbal reasoning:** Models do not perform well on examples requiring interleaved verbal and quantitative reasoning, especially multi-step deduction. Consider the pair ⟨"Two people were injured in the attack", "Two people perpetrated the attack"⟩. Quantities "two people" and "two people" are unit-compatible, but must not be compared. Another example is the NewsNLI entailment pair in Table 1. This pair requires us to identify that 16 and 17 refer to Emmanuel and Zachary's ages (quantitative), deduce that this implies they are teenagers (verbal) and finally count them (quantitative) to get the hypothesis quantity "two teens". Numbers and language are intricately interleaved and developing a reasoner capable of handling such complex interplay is challenging.

2. **Lexical inference:** Lack of real world knowledge causes errors in identifying quantities and valid comparisons. Errors include mapping abbreviations to correct units ("m" to "meters"), detecting part-whole coreference ("seats" can be used to refer to "buses"), and resolving hypernymy/hyponymy ("young men" to "boys").

3. **Inferring underspecified quantities:** Quantity attributes can be implicitly specified, requiring inference to generate a complete representation. Consider "A mortar attack killed four people and injured 80". A system must infer that the quantity "80" refers to people. On RTE-Quant, 20% of such cases stem from zero anaphora, a hard problem in coreference resolution.

4. **Arithmetic comparison limitations:** These examples require composition between incompatible quantities. For example, consider ⟨"There were 3 birds and 6 nests", "There were 3 more nests than birds"⟩. To correctly label this pair "3 birds" and "6 nests" must be composed.

6 Conclusion

In this work, we present EQUATE, an evaluation framework to estimate the ability of models to reason quantitatively in textual entailment. We observe that existing neural approaches rely heavily on the lexical matching aspect of the task to succeed rather than reasoning about quantities. We implement a strong symbolic baseline Q-REAS that achieves success at numerical reasoning, but lacks sophisticated verbal reasoning capabilities. The EQUATE resource presents an opportunity for the community to develop powerful hybrid neuro-symbolic architectures, combining the strengths of neural models with specialized reasoners such as Q-REAS . We hope our insights lead to the development of models that can more precisely reason about the important, frequent, but understudied, phenomena of quantities in natural language.

Acknowledgments

This research was supported in part by grants from the National Science Foundation Secure and Trustworthy Computing program (CNS-1330596, CNS-15-13957, CNS-1801316, CNS-1914486) and a DARPA Brandeis grant (FA8750-15-2-0277). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the NSF, DARPA, or the US Government. The author Naik was supported by a fellowship from the Center of Machine Learning and Health at Carnegie Mellon University. The authors would like to thank Graham Neubig, Mohit Bansal and Dongyeop Kang for helpful discussion regarding this work, and Shruti Rijhwani and Siddharth Dalmia for reviews while drafting this paper. The authors are also grateful to Lisa Carey Lohmueller and Xinru Yan for volunteering their time for pilot studies.

References

Gabor Angeli and Christopher D Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 534–545.

Jorge Balazs, Edison Marrese-Taylor, Pablo Loyola, and Yutaka Matsuo. 2017. Refining raw sentence representations for textual entailment recognition via attention. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 51–55, Copenhagen, Denmark. Association for Computational Linguistics.

Jon Barwise and Robin Cooper. 1981. Generalized quantifiers and natural language. In *Philosophy, Language, and Artificial Intelligence*, pages 241–301. Springer.

Luisa Bentivogli, Elena Cabrio, Ido Dagan, Danilo Giampiccolo, Medea Lo Leggio, and Bernardo Magnini. 2010. Building textual entailment specialized data sets: a methodology for isolating linguistic phenomena relevant to inference. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Languages Resources Association (ELRA).

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 628–635. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017a. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017b. Recurrent neural network-based sentence encoder with gated attention for natural language inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 36–40, Copenhagen, Denmark. Association for Computational Linguistics.

Peter Clark. 2018. What knowledge is needed to solve the rte5 textual entailment challenge? *arXiv preprint arXiv:1806.03561*.

Cleo Condoravdi, Dick Crouch, Valeria De Paiva, Reinhard Stolle, and Daniel G Bobrow. 2003. Entailment, intensionality and text understanding. In *Proceedings of the HLT-NAACL 2003 workshop on Text meaning-Volume 9*, pages 38–45. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. The fourth pascal recognizing textual entailment challenge. *Journal of Natural Language Engineering*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Marie-Catherine De Marneffe, Anna N. Rafferty, and Christopher D. Manning. 2008. Finding contradictions in text. In *Proceedings of ACL-08: HLT*, pages 1039–1047, Columbus, Ohio. Association for Computational Linguistics.

Stanislas Dehaene. 2011. *The number sense: How the mind creates mathematics*. OUP USA.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *CoRR*, abs/1903.00161.

Ruth B Ekstrom, Diran Dermen, and Harry Horace Harman. 1976. *Manual for kit of factor-referenced cognitive tests*, volume 102. Educational Testing Service Princeton, NJ.

Michael C Frank, Daniel L Everett, Evelina Fedorenko, and Edward Gibson. 2008. Number as a cognitive technology: Evidence from pirahã language and cognition. *Cognition*, 108(3):819–824.

Yaroslav Fyodorov. A natural logic inference system. Citeseer.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. Web based probabilistic textual entailment.

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. 2018. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*.

Aria Haghighi, Andrew Ng, and Christopher Manning. 2005. Robust textual inference via graph matching. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

Sanda Harabagiu and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 905–912. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 805–814.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014a. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 271–281.

Nate Kushman, Luke S. Zettlemoyer, Regina Barzilay, and Yoav Artzi. 2014b. Learning to automatically solve algebra word problems. In *ACL*.

Stephen C Levinson. 2001. Pragmatics. In *International Encyclopedia of Social and Behavioral Sciences: Vol. 17*, pages 11948–11954. Pergamon.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017a. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017b. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Bill MacCartney. 2009. *Natural language inference*. Stanford University.

Prodromos Malakasiotis and Ion Androutsopoulos. 2007. Learning textual entailment using svms and string similarity measures. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 42–47. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, Roberto Zamparelli, et al. 2014. A sick cure for the evaluation of compositional distributional semantic models.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2144–2153.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress test evaluation for natural language inference.

In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 41–45, Copenhagen, Denmark. Association for Computational Linguistics.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Lorenza Romano, Milen Kouylekov, Idan Szpektor, Ido Dagan, and Alberto Lavelli. 2006. Investigating a generic paraphrase-based approach for relation extraction.

Subhro Roy. 2017. *Reasoning about quantities in natural language*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Mark Sammons, VG Vydiswaran, and Dan Roth. 2010. Ask not what textual entailment can do for you... In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1199–1208. Association for Computational Linguistics.

Richard E Stafford. 1972. Hereditary and environmental components of quantitative reasoning. *Review of Educational Research*, 42(2):183–201.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 297–306.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

F Zanzotto, Alessandro Moschitti, Marco Pennacchiotti, and M Pazienza. 2006. Learning textual entailment from examples. In *Second PASCAL recognizing textual entailment challenge*, page 50. PASCAL.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822.

Appendix

A Baseline performance on MultiNLI-Dev Matched

Model	MultiNLI Dev
Hyp Only	53.18%
ALIGN	45.0%
CBOW	63.5%
BiLSTM	70.2%
Chen	73.7%
NB	74.2%
InferSent	70.3%
ESIM	76.2%
OpenAI Transformer	81.35%
BERT	83.8%

Table 7: Performance of all baseline models used in the paper on the matched subset of MultiNLI-Dev

Table 7 presents classification accuracies of all baseline models used in this work on the matched subset of MultiNLI-Dev. These scores are very close to the numbers reported by the original publications, affirming the correctness of our baseline setup.

B Examples of quantitative phenomena present in EQUATE

Table 8 presents some examples from EQUATE which demonstrate interesting quantitative phenomena that must be understood to label the pair correctly.

Phenomenon	Example
Arithmetic	**P:** Sharper faces charges in Arizona and California **H:** Sharper has been charged in two states
Ranges	**P:** Between 20 and 30 people were trapped in the casino **H:** Upto 30 people thought trapped in casino
Quantifiers	**P:** Poll: Obama over 50% in Florida **H:** New poll shows Obama ahead in Florida
Ordinals	**P:** Second-placed Nancy celebrated their 40th anniversary with a win **H:** Nancy stay second with a win
Approximation	**P:** Rwanda has dispatched 1917 soldiers **H:** Rwanda has dispatched some 1900 soldiers
Ratios	**P:** Londoners had the highest incidence of E. Coli bacteria (25%) **H:** 1 in 4 Londoners have E. Coli bacteria
Comparison	**P:** Treacherous currents took four lives on the Alabama Gulf coast **H:** Rip currents kill four in Alabama
Conversion	**P:** If the abuser has access to a gun, it increases chances of death by 500% **H:** Victim five times more likely to die if abuser is armed
Numeration	**P:** Eight suspects were arrested **H:** 8 suspects have been arrested
Implicit Quantities	**P:** The boat capsized two more times **H:** His sailboat capsized three times

Table 8: Examples of quantitative phenomena present in EQUATE

Linguistic Analysis Improves Neural Metaphor Detection

Kevin Stowe, Sarah Moeller, Laura Michaelis, Martha Palmer
University of Colorado, Boulder, CO 80309
[kest1439, samo9533, laura.michaelis, mpalmer]@colorado.edu

Abstract

In the field of metaphor detection, deep learning systems are the ubiquitous and achieve strong performance on many tasks. However, due to the complicated procedures for manually identifying metaphors, the datasets available are relatively small and fraught with complications. We show that using syntactic features and lexical resources can automatically provide additional high-quality training data for metaphoric language, and this data can cover gaps and inconsistencies in metaphor annotation, improving state-of-the-art word-level metaphor identification. This novel application of automatically improving training data improves classification across numerous tasks, and reconfirms the necessity of high-quality data for deep learning frameworks.

1 Introduction

Humans use metaphors to conceptualize abstract and often difficult concepts by employing knowledge of more concrete domains. They are prevalent in speech and text, and allow us to communicate more effectively and more imaginatively. The fact that they are commonplace and easily understood by humans makes appropriate interpretation of them essential for high quality natural language processing applications.

The primary linguistic and cognitive theory of metaphor is conceptual metaphor theory (Lakoff and Johnson, 1980; Lakoff, 1993), which theorizes that metaphors are primarily a mental activity, and the language is merely a side effect of these "conceptual" metaphors. From this, it is posited that metaphors are agnostic with regard to syntactic structure: a conceptual mapping can be expressed through whatever syntax the speaker desires. This is apparent from evidence that many metaphoric predications have the same syntactic properties as their literal counterparts. Goldberg (1995) observes that metaphorical ditransitive sentences like "It gave me a headache" do not differ syntactically from literal ditransitive sentences like "She gave me the account." Accordingly, to find syntactic hallmarks of metaphorical meaning we do not look generally for particular syntactic constructions, but rather for mismatches of various kinds between specific verbs' ordinary syntactic behavior and their behavior under metaphoric interpretation.

Perhaps the primary source of verbal syntactic variability is the set of argument-structure constructions identified by Goldberg (1995). One such construction is Caused Motion (CM), illustrated by the sentence "They pushed it down the hall". CM can augment the array of semantic roles supplied by the verb, as in "They laughed me out of the room". Augmentation often entails a metaphoric construal: here the verb "laugh", otherwise a single-argument verb, is paired with both a theme argument (the direct object) and a PP location argument, and the resulting predication expresses metaphorical rather than literal motion (Hwang, 2014).

Despite this connection between verbal syntax and metaphoric properties, most computational approaches to metaphor eschew syntax for more semantic features. While these have proven effective, metaphor detection remains a difficult task. This could be due to many factors, but a primary reason is the lack of adequate training data. Annotation of metaphor has proven to be extremely difficult, as is evident by the variety of schemes used to attempt to achieve consistent annotation.[1] This has led to a lack of "big data" for training models, as well as inconsistencies and gaps in the data that is available.

In this work, we show that syntactic properties

[1]For a review of systems, see Veale et al. (2016)

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 362–371
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

can be used to improve training data, which is beneficial to metaphor processing systems. Deep learning models require sufficient quality data, which is lacking for many metaphorical expressions. We automatically fill gaps in metaphor training data by exploiting syntax in two ways: first, we use the syntactically-motivated lexical resource VerbNet to identify additional data through metaphoric and literal sense identification, and second, we use syntactic properties of certain lexemes, which allow us to identify relevant sentences via dependency parses. These methods yield training data that improves performance for metaphor classification across a variety of tasks.

2 Related Work

While most computational metaphor processing methods rely heavily on lexical semantics, many previous approaches also employ syntactic structures to varying degrees. Most prior work involving argument structure is based on the idea of selectional preferences: certain verbs prefer certain arguments when used literally, and others when used metaphorically. This idea is captured by determining what kinds of arguments fill syntactic and semantic roles for specific verbs.

The CorMet system (Mason, 2004) employs this paradigm, and is similar to ours in their collection of key verbs and analysis of syntactic arguments and semantic roles. They automatically collect documents for particular domains based on key words, and identify selectional preferences based on the WordNet hierarchy for verbs in these particular domains. For example, they find that *assault* typically takes direct objects of the type *fortification* in the *MILITARY* domain. This allows them to make inferences about when selectional preferences are adhered to, and they can then identify mappings between different domains. While their task is fundamentally different, their usage of syntactic frames to identify relevant arguments is very similar to our work. However, rather than identify preferences, we are using syntactic frames to identify whether the verbs are possibly used metaphorically. Our methods require less adherence to semantic properties, which they retrieve from WordNet. Our methods are also inherently somewhat more noisy: while there is evidence that syntactic frames can be indicative of metaphoric properties, these properties are rarely observed deterministically.

Gedigian et al. (2006) use FrameNet and PropBank annotation to collect data, focusing on the FrameNet frames *MOTION* and *CURE*. They use PropBank argument annotations as features, resulting in metaphoric classification accuracy on these domains of over 95%, although this is only slightly above the most frequent class baseline (92%). They collect data from lexical resources and then annotate it for metaphoricity, which is similar to our approach of analyzing the resources and word senses for metaphors.

Shutova et al. (2013) also employs selectional preferences based on argument structure, identifying verb-subject and verb-direct object pairs in corpora. They begin with a seed set of metaphoric pairs, similar to our methods of collecting instances based on syntactic information. They use these seed pairs to identify new metaphors, similar to our usage of syntactic patterns to identify training data. Their methods are based on the selectional preferences of verbs, and thus are less concerned with the variety of syntactic patterns metaphors can participate in. We will identify much more complex syntactic patterns, and we then use the data for training metaphor systems rather than identifying selectional preferences.

Stowe et al. (2018) use syntactic structures directly for feature-based machine learning methods. They highlight the distribution of various syntactic patterns in corpora, and extract features based on dependency parses to improve classifier performance. While their results outperform lexical baselines, they still lag behind other metaphor detection systems, with F1 scores of 53.1 for verbs and 50.5 for nouns on the Vrije Universtat Amsterdam Metaphor Corpus (VUAMC) (Steen et al., 2010). We improve on their work by employing deep learning architecture while still attempting to leverage syntactic information. As many deep learning algorithms (including the recurrent neural networks used here) natively capture long-distance dependencies, direct inclusion of syntactic features is likely not productive. By capturing additional data, we can take advantage of linguistic analysis to improve deep learning-based metaphor detection.

With regard to datasets and tasks, metaphor processing has suffered from a lack of consistent evaluation methods. The metaphor shared task provided a standard evaluation procedure that has greatly helped with system comparison (Leong

et al., 2018). They use the VUAMC, providing a train/test split that has been used to regularize the evaluation of metaphor identification systems. For both sections of the task (identifying all metaphoric words and identifying verbs), four of the top five systems use some form of long short-term memory network (LSTM). The system of Wu et al. (2018) performed best on both tasks (F1 of .651 on all parts of speech, and .672 for verbs) using a combination of a convolutional neural network (CNN) and bidirectional LSTM.

Since the shared task, a variety of other approaches have been developed using similar deep learning techniques. Most recently, the work of Gao et al. (2018) achieved state-of-the-art performance on the shared task data as well as a variety of other datasets, including the TroFi (Birke and Sarkar, 2006) and Mohammad et al. (2016) datasets, using Bi-LSTM models coupled with GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018) embeddings. For the VUAMC shared task, they report F1 scores of .726 for all parts of speech and .697 for verbs. For the Mohammad et al. dataset, they report an average F1 score of .791, and for the Trofi they report an F1 score of .72, slightly lower than the current state of the art (.75 of Köper et al. (2017)).

Despite recent advances in evaluation and algorithm performance, the task still remains difficult, with the highest F1 scores nearing only .73 on the VUAMC data. This is likely due to the relatively small dataset size (app. 200,000 words), which is in part caused by difficulties in annotation. We aim to overcome some of this difficulty by automatically extracting additional training data with lexical and syntactic methods.

3 Methods

We aim to improve training data for metaphor processing by performing linguistic analysis on difficult verbs to uncover the syntactic properties that can potentially influence their metaphoricity. Our work is focused on verbs: they form the foundation of many metaphoric expressions, and evidence from construction grammar and frame semantics has shown that syntactic properties can often influence the types of metaphors that are produced (Sullivan, 2013). We show that identification of anomalous syntactic structures can provide evidence towards metaphoricity, and can be leveraged to automatically extract training data that improves classification performance.

This is done through two paths: first, we explore the lexical semantic resource VerbNet, an ontology of English verbs that contains rich syntactic and semantic information. From VerbNet we explore verb senses that can potentially be deterministically metaphoric or literal, and extract training data from existing VerbNet annotation. Second, we analyze syntactic patterns from Wikipedia data. We identify patterns that indicate metaphoric or literal senses of verbs, and then extract additional data based on these patterns.

3.1 Finding Difficult Verbs

First, we need to select verbs to analyze. Our goal is to find verbs that are likely difficult for classifiers, as well as those that are frequent enough to have a significant impact. This is accomplished through two avenues: first, we examine all the verbs in the training data, and analyze those that have the most even class balance between literal and metaphoric uses. We refer to these verbs as our most "ambiguous" verbs. For our preliminary experiments, we selected the ten most ambiguous verbs which occurred at least ten times in the training data.

Second, we employed the metaphor detection system of Gao et al. (2018). We trained the system on the provided VUAMC shared task training data and ran it on their validation set. We then analyzed which verbs were most frequently misclassified in the validation data, to determine where additional data would be most effective. We chose to use the VUAMC data for this task due to its size and status as the standard for metaphor identification. As with the most ambiguous verbs, we selected the ten verbs with the lowest F1 score that occurred at least ten times in the data for analysis. The verbs chosen through these analyses are shown in Table 1. In theory, expanding the number of verbs analyzed would yield more data and improve performance, but as an experimental baseline ten verbs is sufficient for analysis and classifier improvement.

For each of these verbs, we performed two kinds of analysis. First, we explored their metaphoric and literal usage in VerbNet. Second, we examined their syntactic properties for metaphoric and literal patterns.

3.2 VerbNet

VerbNet is a lexical resource that currently categorizes 6,791 verbs into 329 verb classes based

Most Ambiguous Verbs				Most Misclassified Verbs				
Verb	**Met Tokens**	**Lit Tokens**	**% Met**	**Verb**	**FP**	**FN**	**Correct**	**% Correct**
encourage	6	6	.5	spend	7	0	4	.363
blow	5	5	.5	include	8	1	10	.526
conduct	5	5	.5	play	3	3	8	.571
show	34	33	.49	hold	3	3	8	.571
find	60	62	.49	stop	7	1	12	.600
fall	18	19	.51	reduce	2	2	6	.600
hold	28	30	.52	get	15	21	74	.673
bring	36	33	.48	suggest	4	0	9	.692
put	57	52	.48	meet	2	1	7	.700
allow	19	21	.48	discuss	1	2	7	.700

Table 1: Difficult Verbs from the VUAMC data: on the left, the verbs with the most even split between literal and metaphoric. On the right, verbs in the validation set that were most misclassified. Restricted to verbs where count ≥ 10.

on their syntactic and semantic behavior (Kipper-Schuler, 2005).[2] These verb classes are based on the work of Levin (1993), who shows that for many verbs their semantics can be determined by the syntactic alternations they participate in, arguing that "the behavior of a verb, particularly with respect to the expression and interpretation of its arguments, is to a large degree determined by its meaning." (pg 1)

VerbNet is primarily composed of verb "classes": these classes are a hierarchically structuring of verb senses based on their syntactic and semantic behavior. Each class contains a list of verb senses, the syntactic frames that these verbs can participate in, a first-order semantic predicate representation for the class's meaning, and the thematic roles the verb takes as arguments. These thematic roles, which are fairly coarse-grained roles such as Agent, Theme, and Patient, are often marked with selectional restrictions. For example, many classes have Agents that are marked as +ANIMATE, indicating the Agent of the verb must be an animate entity.

VerbNet has practical applications for word sense disambiguation and semantic role labelling, and numerous annotation projects have been done to tag data with the correct VerbNet senses. Our goal is to identify which particular VerbNet senses are typically metaphoric or literal, and extract sentences tagged with these VerbNet senses.

For each verb, we examined the VerbNet classes in which it appears. We looked at VerbNet annotation, the example sentences, the selectional preferences on the class's thematic roles, and the semantic predicates. From this we assessed whether the sense of the verb in each class was typically metaphoric or literal. Consider the verb "grow". It is present in two particular VerbNet classes: **GROW-26.2** and **CALIBRATABLE_COS-45.6**. The **GROW-26.2** class has an animate Agent role, and produces a concrete Product out of a concrete Material:

1. A private farmer in Poland is free to buy and sell land, hire help, and decide what to **grow**.

2. It's the kind of fruit that **grew** freely and that you could help yourself to.

We note from the semantics and annotated examples, we expect this sense of grow to typically be literal. However, in the **CALIBRATABLE_COS-45.6** class, it contains a Value role that moves along a scale by a certain Extent. These examples all appear to be metaphoric, evoking the MORE IS UP mapping[3]:

1. Exports in the first eight months **grew** only 9%.

2. Non-interest expenses **grew** 16% to $496 million.

This allows us to extract new training data using these classes. We use a repository of manually annotated VerbNet senses, containing approximately 150,000 annotated verbs (Palmer et al.,

[2] https://verbs.colorado.edu/verbnet/

[3] Examples from the VerbNet annotation data from Palmer et al. (2017)

2017). We found all annotated instances of "grow" in the **GROW-26.2** class, and considered them to be literal, and all instances of "grow" from **CALIBRATABLE_COS-45.6** were considered metaphoric. This process was completed for all of the verbs in Table 1. This analysis only includes the particular verb in question. We believe "grow" in the **GROW-26.2** class is typically literal, and "grow" in the **CALIBRATABLE_COS-45.6** class is typically metaphoric, but this does not necessarily extend to other verbs in these classes. We will discuss the possibility of expanding this analysis to include all verbs for particular classes in Section 6.

Note that we only consider the verbs in these instances: we have no knowledge of the metaphoricity of the verbs' arguments. For each verb, we extracted up to 100 annotations for each sense that we determined to be largely metaphoric or literal.

3.3 Syntactic Patterns

As a second path for finding additional data, we explore the syntactic properties of metaphoric expressions. While metaphors are traditionally seen as cognitive, and relatively unaffected by surface syntactic realizations, there is recent evidence based in construction grammar that syntactic structures can influence the source and target domain elements of metaphoric expressions (Sullivan, 2013; David and Matlock, 2018; David, 2017). We expand on this idea: we believe that not only can syntactic structures indicate source and target elements, but they can also indicate metaphoricity.

We see this in English with verbs like hemorrhage, which is almost always used metaphorically when it is used transitively[4]:

- GM was supporting this event even as they were **hemorrhaging cash**.
- For 30 straight years, American organized labor has been **hemorrhaging members**.

When used intransitively, hemorrhage is almost always literal:

- Cerebral AVMs often have no symptoms until they rupture and **hemorrhage**.
- Michael **hemorrhaged** and sustained a massive stroke to the left side of his brain.

[4]Examples from SketchEngine (Kilgarriff et al., 2014) http://www.sketchengine.eu/

This is likely due to the fact that literal use of "hemorrhage" contains an understood argument, blood, which is the most natural object of the verb. If the use is intended in a less literal way, which requires on over syntactic object, the null "blood" object needs to be overridden. While not all verbs have this direct relation between argument number and metaphoricity, we believe that the type and number of syntactic arguments of a verb can be indicative of unmarked usage, and may be utilized as a method for automatically extracting training data for metaphor classification. This analysis doesn't reflect linguistic facts: it is possible to construct sentences in which the intransitive use of "hemorrhage" is metaphoric ("after the stock market crashed, the company hemorrhaged"), as well as transitive usages that are literal ("after the surgery, the patient hemorrhaged blood"). However, we find that in the majority of cases, metaphoricity aligns with the argument structure, and these contrived examples are exceedingly rare.

For each verb in our list, we analyzed all the sentences from the VUAMC training data as well as 50 additional sentences from Wikipedia that contained the verb, and attempted to discover syntactic patterns that are indicative of metaphoricity. We examined argument structure, active vs passive voice, prepositional complements, aspect, idiomatic combinations and other surface syntactic properties. We created a short list of the most likely candidates for literal and metaphoric syntactic patterns. We then extracted up to 100 sentences from Wikipedia that matched these syntactic patterns.

A brief overview of the syntactic patterns and VerbNet class analysis is shown in Table 2; the full extraction rules, code, and data will be released upon publication.

Note that for many cases, it was difficult to determine what was literal and what was metaphoric. Highly polysemous verbs like "get" in particular are problematic: they contain many different meanings and usages that can often be annotated inconsistently, so strong metaphoric or literal patterns were impossible to identify.

As with the "hemorrhage" examples above, these patterns are not deterministic. The syntactic structures analyzed aren't always metaphoric or literal, but they are consistent enough to be useful for extracting additional training data. For each verb we attempted to extract up to 100 samples

Verb	Lit. Syn. Patterns	Met. Syn. Patterns	Lit. VerbNet Classes	Met. VerbNet Classes
encourage	NP V NP {TO} VP	NP V NP	advise-37.9	amuse-31.1
find	NP V PRO VP *find out*, *find dead*	NP V NP {TO BE} ADJ	get-13.5.1	declare-29.4
fall	NP V ADV, NP V	WH NP V *fall in*, *fall to*	escape-51.1	calibratable_cos-45.6 convert-26.6.2 long-32.2 acquiesce-95.1 die-42.4
spend	NP V NP V {ON} NP	*spend time* *spend life*	pay-68	consume-66 spend_time-104
play	NP V PP *play with*	-	meet-36.3 performance-36.7 play-114.2	trifle-105.3 use-105.1
suggest	negation	-	say-37.7	reflexive_appearance-48.1.2
meet	NP V *meet for/at/to*	-	contiguous_location-47.8	satisfy-55.7

Table 2: Example analysis of syntactic patterns and VerbNet classes.

	From VerbNet		From Syn. Patterns	
Verb	Count	% Met	Count	% Met
encourage	86	.611	200	.497
blow	-	-	99	.946
conduct	-	-	200	.503
show	-	-	-	-
find	407	.300	255	.047
fall	314	.601	600	.749
hold	913	.445	487	.560
bring	-	-	500	.601
put	-	-	-	-
allow	2	1	300	.334
spend	439	.630	341	.553
play	52	.196	343	0
stop	482	.208	-	-
reduce	-	-	-	-
suggest	307	.003	12	0
meet	455	.229	399	0
Total	3985	.442	3736	.424

Table 3: Total samples extracted from VerbNet classes and syntactic patterns, along with the percentage of extracted samples that are metaphoric.

for each VerbNet sense and each syntactic pattern. This is to prevent the extracted data becoming saturated with extremely common senses or patterns. Many senses and patterns are rare, and fewer than 100 instances were collected. A summary of the extracted data by verb is shown in Table 3.

In total, we extracted 3,985 samples from VerbNet annotation and 3,736 samples from Wikipedia based on syntactic samples for our analyzed verbs. Each sample is an entire sentence containing the verb in question, for which we can provide automatic annotation based on our VerbNet and syntactic analyses. We can treat this as distantly supervised data: we have beliefs about the metaphoric and literal labels for the verbs in each sentence extracted, but these aren't always deterministic: errors in syntactic pattern matching, anomalous examples, and other factors introduce inaccuracies in these samples.

4 Tasks

In order to show the efficacy of our extracted data, we add this data to the standard datasets and evaluate performance on a variety of metaphor processing tasks. For a relevant comparison to contemporary research, we evaluate our results using the baseline system of Gao et al. on five different tasks. As per their work, we experiment with two different models: a sequence based model (dubbed "SEQ") that performs best when all parts of speech contain metaphor tags, and a "classification" model ("CLS"), which tags individual verbs as metaphoric or not.

4.1 Sequential Model (SEQ)

The sequential model takes as input sentences from VUAMC data, each with a binary metaphor tag. They represent each word as the concatenation of a 300 dimension GloVe embeddings with an ELMo vector. These are then input to a bidirectional LSTM. These sequential models are particularly useful for encoding relations among distant words, and have proven effective on a large number of tasks for which each word in a sentence has a tag.

4.2 Classification Model (CLS)

The classification model represents each verb in the VUAMC data as its own instance, maintaining the sentential context, and these each retain their annotation as either metaphoric or not. As with the

sequential model, they are input to a bidirecitonal LSTM using GloVe and ELMo vectors. They also include an index embedding and attention layer to encode the location of the target word.[5]

These models are used over a variety of datasets: the dataset from Mohammad et al. (2016), the Trofi dataset (Birke and Sarkar, 2006), and the VUA metaphor corpus which is the basis for the metaphor shared task (Steen et al., 2010). They use a section of the Mohammad et al. dataset dubbed "MOH-X", consisting of 636 example sentences from 214 verbs taken from WordNet, annotated as metaphoric or literal. The Trofi dataset contains 3,737 sentences from 50 different verbs which were automatically clustered into metaphoric or literal clusters. The sequence tagger shows best performance when all words in the sentence are metaphoric, as is the case with the VUAMC data. The classification model performs best when only a single word is metaphoric, as in the MOH-X and Trofi datasets. While the VUAMC is the basis of our analysis, we will also examine how adding additional impacts results on the MOH-X and Trofi datasets; their setup as classification tasks more accurately mirrors the additional data, as there is only one potentially metaphoric word per sample.

4.3 Architecture

We replicate the architectures of Gao et al., using the same experimental set-up. For the classification model, we can include our extra data as-is, with metaphor annotations based on our analysis. For the sequential model, we consider only the verb analyzed as metaphoric, leaving the rest of the words tagged as literal. In order to judge performance, we run three experimental setups: one with the additional VerbNet samples, one with the additional samples generated via syntactic patterns, and one with both. We experimented with tuning hyperparameters (learning rate, dropout, and the size of the hidden layer), but found no significant improvements over their experimental setup. We did make one modification: we increased the amount of training epochs in proportion to the amount of training data added. This allows for the model to be sufficiently trained over all the data.

The VUAMC data has been split into training and test sections for the shared task, and these sections are also used by Gao et al. We will adopt this split. For the MOH-X and Trofi datasets, they run 10-fold cross validation and report the mean F1 score. Due to the variable nature of neural models and the relatively small dataset size, we include experiments to calculate the statistical significance of our methods. We split these datasets into 75% training, 25% test, mirroring the VUAMC data, and ran classification 10 times. We then calculated the means and standard deviations. We also ran bootstrap estimation for all tasks, reevaluating using random replacement over 10^6 iterations (Efron, 1979; Berg-Kirkpatrick et al., 2012). We consider improvement significant when the mean and standard deviation from both methods yield p values of less than .01.

5 Results

The results from our additions on the original tasks are shown in Figure 1, and the improvements over the baseline for each method are outlined in Table 4. For each task, we display the results of the original Gao baseline, along with the addition of VerbNet samples, syntactic pattern-based samples, and both. For each of these, we show the mean and standard deviation from running the task 10 times.

We find that adding VerbNet samples, syntactic patterns, and both datasets all always produces a significant improvement over the baseline. Adding this additional data outperforms the Gao et al. sequence tagging algorithm on the VUA shared task data for both verbs and all parts of speech. We also see improvements in the classification model, and on the Trofi dataset. It is important to note that the extreme variability in the results for these smaller datasets. We found our improvements on the Trofi dataset to be significant, while the MOH-X results were not significant. This is likely due to the size of the dataset: the MOH-X data contains only 636 samples, leading to high variance in performance. Further evaluation is necessary to determine the consistent effect of this data.

For verb sequence tagging, the VerbNet data yielded the best performance, while for all parts of speech the additional syntactic data performed best. This may be because the VerbNet data comes specifically from VerbNet annotation, relying strictly on VerbNet senses. VerbNet is grounded in syntactic alternations, but individually VerbNet senses occasionally encode metaphor

[5]Full details and code for each of these models can be found at https://github.com/gao-g/metaphor-in-context/

	Task				
Additional Data	MOH-X	Trofi	VUA CLS (Verbs)	VUA SEQ (Verbs)	VUA SEQ (All)
Baseline	.653	.658	.665	.682	.728
+VN Data	.681*	.672	.673	**.696**	.736
+SYN Data	.704*	.672	.677	.694	**.738**
+Both	.683*	**.684**	**.679**	.695	.735

Table 4: Mean F1 scores over 10 iterations, for each model and dataset added. Bootstrap sampling indicated that these improvements are all significant over the baseline, excepting the MOH-X dataset. Due to high variability, the MOH-X results (*) were not significant improvements over the baseline ($p > .01$)

.

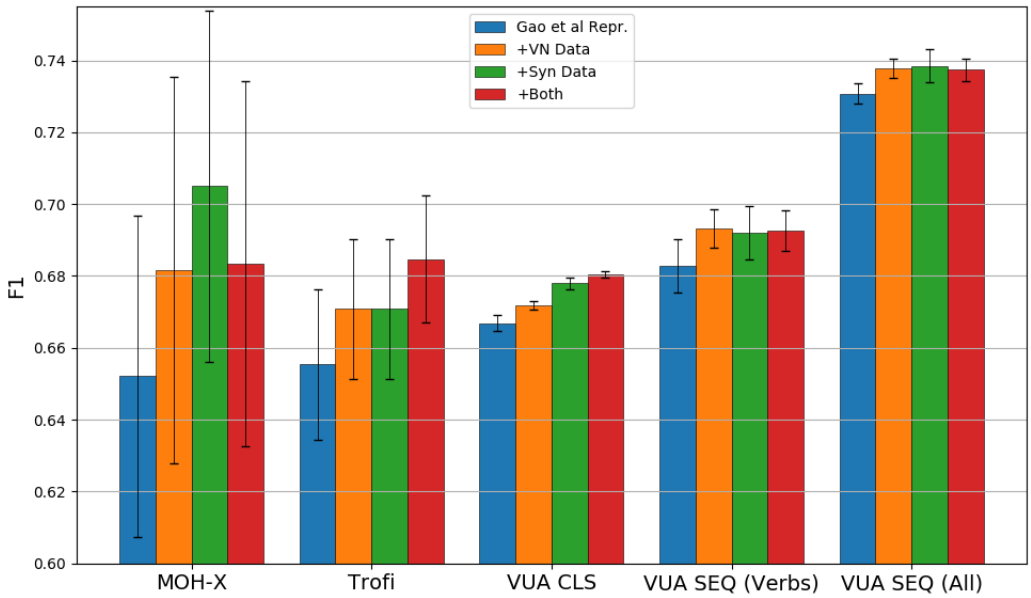


Figure 1: Results for each task. Results shown are the mean with standard deviations from running 10 iterations of each model for each task.

without direct relation to the verb's arguments. The syntactic data directly relies on patterns which include other parts of speech: arguments, prepositions, and idiomatic expressions. These extra components of the analysis may make the data more broadly applicable to all parts of speech, driving the improvement in the sequence tagging of all words.

Adding both distantly supervised datasets improved performance over adding either individually only for the classification-based tasks, where only one word per sentence has a tag (the Trofi and VUAMC verb classification tasks). Only on the Trofi dataset was the improvement from adding both datasets significantly higher than the improvement from adding the best individual dataset. For the sequence-based tagging of the VUAMC, adding both yielded negligible improvements. As adding both datasets was effective for classification tasks, we believe the difficulty in combining both datasets in the sequence models is due to excessive noise from the non-target words of the samples. We default to marking every word other than the target verb in the sentence as literal, so the additional data is understandably less informative for sequence tagging problems. It is likely that the combination of VerbNet data and syntactic pattern-based data caused additional noise: the two datasets may in places be contradictory, particularly with regard to these non-target elements.

6 Conclusions

We show that using external data found through syntactic structures and lexical resources can be used to improve deep learning methods for metaphoric classification. This is due to regular syntactic patterns of metaphoric usage, and the idea that the semantics of verbs can be dependent on the syntactic patterns that it participates in. For future improvements, there are other resources available that could be leveraged in the same way. PropBank (Palmer et al., 2005), FrameNet (Baker

et al., 1998), and WordNet (Fellbaum, 2010) all offer some syntactic and/or semantic information, and data annotated with these resources could prove another valuable source of additional samples.

We also only examine some basic syntactic patterns for a small number of verbs, and this was done manually. Improved methods for automatically detecting relevant syntactic patterns as well as further effort in manual identification of syntactic properties of metaphoric samples could increase the amount of data extracted. Further linguistic analysis of constructions that either require or prohibit metaphoric interpretations could improve both automatic metaphor processing and our broader understanding of linguistic metaphors. Additionally, we only look at specific verbs within VerbNet classes. All verbs within VerbNet classes share syntactic and semantic properties, so it is likely that we can extend our verb-level analysis to a broader class-level analysis. A straightforward extension of this work would be to analyze VerbNet classes as being metaphoric or literal, and extracting data for all verbs within a given class.

Finally, while they have proven invaluable for the standardization of metaphor processing, there are still gaps and inconsistencies in our metaphor datasets. Extracting additional training data based on syntactic patterns likely was effective in this case in part due to the idiosyncrasies of the previous datasets, which may over-annotate possible metaphors. This procedure yields a large number of conventional metaphors, which lack novelty, are very frequent, and are perhaps more amenable to being discovered via syntactic patterns. More data annotated for metaphor is essential to improve deep learning methods for metaphor processing, and while we are attempting to overcome these gaps with outside resources, further quality metaphor annotation would prove especially valuable to the field.

Acknowledgements

We gratefully acknowledge the support of the Defense Threat Reduction Agency, HDTRA1-16-1-0002/Project #1553695, eTASC - Empirical Evidence for a Theoretical Approach to Semantic Components and a grant from the Defense Advanced Research Projects Agency 15-18-CwC-FP-032 Communicating with Computers, a subcontract from UIUC. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any government agency.

References

C. F. Baker, C.J. Fillmore, and J.B. Lowe. 1998. The Berkeley FrameNet project. In *COLING-ACL '98*, pages 86–90, Montreal, QC.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. "An Empirical Investigation of Statistical Significance in NLP". In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005. Association for Computational Linguistics.

Julia Birke and Anoop Sarkar. 2006. "A Clustering Approach for Nearly Unsupervised Recognition of Nonliteral Language". In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Oana David. 2017. "Computing metaphor: The case of MetaNet". *Cambridge Handbook of Cognitive Linguistics*, pages 574–589.

Oana David and Teenie Matlock. 2018. Cross-linguistic automated detection of metaphors for poverty and cancer. *Language and Cognition*, 10(3):467–593.

Bradley Efron. 1979. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*.

George A. Fellbaum, Christiane; Miller. 2010. WordNet. `http://wordnet.princeton.edu/`. Accessed: 2019-02-28.

Ge Gao, Eunsol Choi, Yejin Choi, and Luke Zettlemoyer. 2018. "Neural Metaphor Detection in Context". In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 607–613. Association for Computational Linguistics.

Matt Gedigian, John Bryant, Srini Narayanan, and Branimir Ciric. 2006. Catching metaphors. In *Proceedings of the Third Workshop on Scalable Natural Language Understanding*, ScaNaLU '06, pages 41–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Adele Goldberg. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press.

Jena Hwang. 2014. *"Identification and Representation of Caused-Motion Constructions"*. Ph.D. thesis, University of Colorado, Boulder.

Adam Kilgarriff, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlý, and Vít Suchomel. 2014. The sketch engine: ten years on. *Lexicography*, pages 7–36.

Karen Kipper-Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Maximilian Köper and Sabine Schulte im Walde. 2017. "Improving Verb Metaphor Detection by Propagating Abstractness to Words, Phrases and Individual Senses". In *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*, pages 24–30, Valencia, Spain. Association for Computational Linguistics.

George Lakoff. 1993. The Contemporary Theory of Metaphor. In Andrew Ortony, editor, *Metaphor and Thought*, pages 202–251. Cambridge University Press.

George Lakoff and Mark Johnson. 1980. *Metaphors We Live By*. University of Chicago Press, Chicago and London.

Chee Wee (Ben) Leong, Beata Beigman Klebanov, and Ekaterina Shutova. 2018. "A Report on the 2018 VUA Metaphor Detection Shared Task". In *Proceedings of the Workshop on Figurative Language Processing*, pages 56–66. Association for Computational Linguistics.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. The University of Chicago Press.

Zachary J. Mason. 2004. "CorMet: A computational, corpus-based conventional metaphor extraction system". *Computational Linguistics*, 30(1):23–44.

Saif Mohammad, Ekaterina Shutova, and Peter Turney. 2016. "Metaphor as a Medium for Emotion: An Empirical Study". In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 23–33. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Association of Computational Linguitics*, 31(1):71–106.

Martha Palmer, James Gung, Claire Bonial, Jinho Choi, Orin Hargraves, Derek Palmer, and Kevin Stowe. 2017. The Pitfalls of Shortcuts: Tales from the word sense tagging trenches. In *Essays in Lexical Semantics and Computational Lexicography - In honor of Adam Kilgarriff*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Ekaterina Shutova, Simone Teufel, and Anna Korhonen. 2013. Statistical metaphor processing. *Computational Linguistics*, 39(2):301–353.

Gerard J. Steen, Aletta G. Dorst, J. Berenike Herrmann, Anna A. Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A Method for Linguistic Metaphor Identification*. John Benjamins Publishing Company.

Kevin Stowe and Martha Palmer. 2018. Leveraging Syntactic Constructions for Metaphor Processing. In *Workshop on Figurative Language Processing*, New Orleans, Louisiana.

Karen Sullivan. 2013. *Frames and Constructions in Metaphoric Language*. John Benjamins.

T. Veale, E. Shutova, and B. Klebanov. 2016. *Metaphor: A Computational Perspective*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Chuhan Wu, Fangzhao Wu, Yubo Chen, Sixing Wu, Zhigang Yuan, and Yongfeng Huang. 2018. "Neural Metaphor Detecting with CNN-LSTM Model". In *Proceedings of the Workshop on Figurative Language Processing*, pages 110–114. Association for Computational Linguistics.

Cross-lingual Dependency Parsing with Unlabeled Auxiliary Languages

Wasi Uddin Ahmad[1], **Zhisong Zhang**[2], **Xuezhe Ma**[2]
wasiahmad@cs.ucla.edu,{zhisongz,xuezhem}@cs.cmu.edu
Kai-Wei Chang[1], **Nanyun Peng**[3]
kwchang@cs.ucla.edu,npeng@isi.edu

[1]University of California, Los Angeles, [2]Carnegie Mellon University
[3]University of Southern California

Abstract

Cross-lingual transfer learning has become an important weapon to battle the unavailability of annotated resources for low-resource languages. One of the fundamental techniques to transfer across languages is learning *language-agnostic* representations, in the form of word embeddings or contextual encodings. In this work, we propose to leverage unannotated sentences from auxiliary languages to help learning language-agnostic representations. Specifically, we explore adversarial training for learning contextual encoders that produce invariant representations across languages to facilitate cross-lingual transfer. We conduct experiments on cross-lingual dependency parsing where we train a dependency parser on a source language and transfer it to a wide range of target languages. Experiments on 28 target languages demonstrate that adversarial training significantly improves the overall transfer performances under several different settings. We conduct a careful analysis to evaluate the language-agnostic representations resulted from adversarial training.

1 Introduction

Cross-lingual transfer, where a model learned from one language is transferred to another, has become an important technique to improve the quality and coverage of natural language processing (NLP) tools for languages in the world. This technique has been widely applied in many applications, including part-of-speech (POS) tagging (Kim et al., 2017), dependency parsing (Ma and Xia, 2014), named entity recognition (Xie et al., 2018), entity linking (Sil et al., 2018), coreference resolution (Kundu et al., 2018), and question answering (Joty et al., 2017). Noteworthy improvements are achieved on low resource language applications due to cross-lingual transfer learning.

In this paper, we study cross-lingual transfer for dependency parsing. A dependency parser consists of (1) an encoder that transforms an input text sequence into latent representations and (2) a decoding algorithm that generates the corresponding parse tree. In cross-lingual transfer, most recent approaches assume that the inputs from different languages are aligned into the same embedding space via multilingual word embeddings or multilingual contextualized word vectors, such that the parser trained on a source language can be transferred to target languages. However, when training a parser on the source language, the encoder not only learns to embed a sentence but it also carries language-specific properties, such as word order typology. Therefore, the parser suffers when it is transferred to a language with different language properties. Motivated by this, we study how to train an encoder for generating language-agnostic representations that can be transferred across a wide variety of languages.

We propose to utilize *unlabeled* sentences of one or more auxiliary languages to train an encoder that learns language-agnostic contextual representations of sentences to facilitate cross-lingual transfer. To utilize the unlabeled auxiliary language corpora, we adopt adversarial training (Goodfellow et al., 2014) of the encoder and a classifier that predicts the language identity of an input sentence from its encoded representation produced by the encoder. The adversarial training encourages the encoder to produce language invariant representations such that the language classifier fails to predict the correct language identity. As the encoder is jointly trained with a loss for the primary task on the source language and adversarial loss on all languages, we hypothesize that it will learn to capture task-specific features as well as generic structural patterns applicable to many languages, and thus have better transferrability.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 372–382
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

To verify the proposed approach, we conduct experiments on neural dependency parsers trained on English (source language) and directly transfer them to 28 target languages, with or without the assistance of unlabeled data from auxiliary languages. We chose dependency parsing as the primary task since it is one of the core NLP applications and the development of Universal Dependencies (Nivre et al., 2016) provides consistent annotations across languages, allowing us to investigate transfer learning in a wide range of languages. Thorough experiments and analyses are conducted to address the following research questions:

- Does encoder trained with adversarial training generate language-agnostic representations?
- Does language-agnostic representations improve cross-language transfer?

Experimental results show that the proposed approach consistently outperform a strong baseline parser (Ahmad et al., 2019), with a significant margin in two family of languages. In addition, we conduct experiments to consolidate our findings with different types of input representations and encoders. Our experiment code is publicly available to facilitate future research.[1]

2 Training Language-agnostic Encoders

We train the encoder of a dependency parser in an *adversarial* fashion to guide it to avoid capturing language-specific information. In particular, we introduce a language identification task where a classifier predicts the language identity (id) of an input sentence from its encoded representation. Then the encoder is trained such that the classifier fails to predict the language id while the parser decoder predicts the parse tree accurately from the encoded representation. We hypothesize that such an encoder would have better cross-lingual transferability. The overall architecture of our model is illustrated in Figure 1. In the following, we present the details of the model and training method.

2.1 Architecture

Our model consists of three basic components, (1) a general encoder, (2) a decoder for parsing, and (3) a classifier for language identification. The encoder learns to generate contextualized representations for the input sentence (a word sequence)

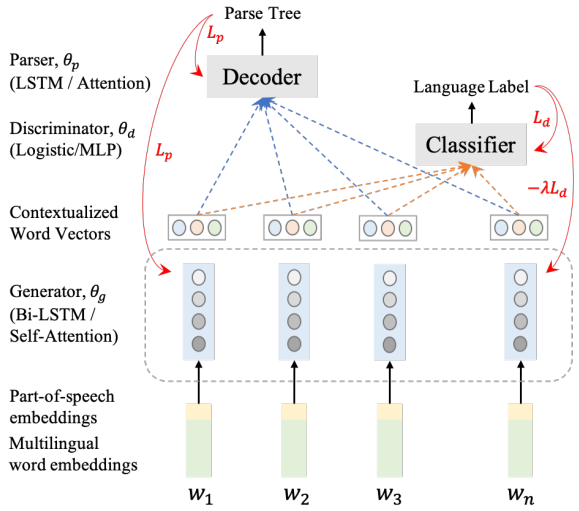


Figure 1: An overview of our experimental model consists of three basic components: (1) Encoder, (2) (Parsing) Decoder, and (3) (Language) Classifier. We also show how parsing and adversarial losses (L_p and L_d) are back propagated for parameter updates.

which are fed to the decoder and the classifier to predict the dependency structure and the language identity (id) of that sentence.

The encoder and the decoder jointly form the parsing model and we consider two alternatives[2] from (Ahmad et al., 2019): "SelfAtt-Graph" and "RNN-Stack". The "SelfAtt-Graph" parser consists of a modified self-attentional encoder (Shaw et al., 2018) and a graph-based deep bi-affine decoder (Dozat and Manning, 2017), while the "RNN-Stack" parser is composed of a Recurrent Neural Network (RNN) based encoder and a stack-pointer decoder (Ma et al., 2018).

We stack a classifier (a linear classifier or a multi-layer Perceptron (MLP)) on top of the encoder to perform the language identification task. The identification task can be framed as either a word- or sentence-level classification task. For the sentence-level classification, we apply average pooling[3] on the contextual word representations generated by the encoder to form a fixed-length representation of the input sequence, which is fed to the classifier. For the word-level classification, we perform language classification for each token individually.

[1] https://github.com/wasiahmad/cross_lingual_parsing

[2] Ahmad et al. (2019) studied order-sensitive and order-free models and their performances in cross-lingual transfer. In this work, we adopt two typical ones and study the effects of adversarial training on them.

[3] We also experimented with max-pooling and weighted pooling but average pooling resulted in stable performance.

Algorithm 1 Training procedure.

Parameters to be trained: Encoder (θ_g), Decoder (θ_p), and Classifier (θ_d)
X^a = Annotated source language data
X^b = Unlabeled auxiliary language data
I = Number of warm-up iterations
k = Number of learning steps for the discriminator (D) at each iteration
λ = Coefficient of $\mathcal{L}_d$
α_1, α_1 = learning rate; B = Batch size

Require:

for $j = 0, \cdots, I$ **do**
  Update $\theta_g := \theta_g - \alpha_1 \nabla_{\theta_g} \mathcal{L}_p$
  Update $\theta_p := \theta_p - \alpha_1 \nabla_{\theta_p} \mathcal{L}_p$
for $j = I, \cdots, num_iter$ **do**
  for k steps **do**
    $(x_a^i)_{i=1}^{B/2} \leftarrow$ Sample a batch from X^a
    $(x_b^i)_{i=1}^{B/2} \leftarrow$ Sample a batch from X^b
    Update $\theta_d := \theta_d - \alpha_2 \nabla_{\theta_d} \mathcal{L}_d$
  Total loss $\mathcal{L} := \mathcal{L}_p - \lambda \mathcal{L}_d$
  Update $\theta_g := \theta_g - \alpha_1 \nabla_{\theta_g} \mathcal{L}$
  Update $\theta_p := \theta_p - \alpha_1 \nabla_{\theta_p} \mathcal{L}$

In this work, following the terminology in adversarial learning literature, we interchangeably call the encoder as the generator, G and the classifier as the discriminator, D.

2.2 Training

Algorithm 1 describes the training procedure. We have two types of loss functions: $\mathcal{L}_p$ for the parsing task and $\mathcal{L}_d$ for the language identification task. For the former, we update the encoder and the decoder as in the regular training of a parser. For the latter, we adopt adversarial training to update the encoder and the classifier. We present the detailed training schemes in the following.

2.2.1 Parsing

To train the parser, we adopt both cross-entropy objectives for these two types of parsers as in (Dozat and Manning, 2017; Ma et al., 2018). The encoder and the decoder are jointly trained to optimize the probability of the dependency trees (y) given sentences (x):

$$\mathcal{L}_p = -\log p(y|x).$$

The probability of a tree can be further factorized into the products of the probabilities of each token's (m) head decision ($h(m)$) for the graph-based parser, or the probabilities of each transition step decision (t_i) for the transition-based parser:

$$\text{Graph:} \quad \mathcal{L}_p = -\sum_m \log p(h(m)|x, m),$$
$$\text{Transition:} \quad \mathcal{L}_p = -\sum_i \log p(t_i|x, t_{<i}).$$

2.2.2 Language Identification

Our objective is to train the contextual encoder in a dependency parsing model such that it encodes language specific features as little as possible, which may help cross-lingual transfer. To achieve our goal, we utilize adversarial training by employing unlabeled auxiliary language corpora.

Setup We adopt the basic generative adversarial network (GAN) for the adversarial training. We assume that X^a and X^b be the corpora of the source and auxiliary language sentences, respectively. The discriminator acts as a binary classifier and is adopted to distinguish the source and auxiliary languages. For the training of the discriminator, weights are updated according to the original classification loss:

$$\mathcal{L}_d = \mathbb{E}_{x \sim X^a}[\log\ D(G(x)) + \mathbb{E}_{x \sim X^b}[\log\ (1 - D(G(x))].$$

For the training of dependency parsing, the generator, G collaborates with the parser but acts as an adversary with respect to the discriminator. Therefore, the generator weights (θ_g) are updated by minimizing the loss function,

$$\mathcal{L} = \mathcal{L}_p - \lambda \mathcal{L}_d,$$

where λ is used to scale the discriminator loss ($\mathcal{L}_d$). In this way, the generator is guided to build language-agnostic representations in order to fool the discriminator while being helpful for the parsing task. Meanwhile, the parser can be guided to rely more on the language-agnostic features.

Alternatives We also consider two alternative techniques for the adversarial training: Gradient Reversal (GR) (Ganin et al., 2016) and Wasserstein GAN (WGAN) (Arjovsky et al., 2017). As opposed to GAN based training, in GR setup, the discriminator acts as a multiclass classifier that predicts language identity of the input sentence, and we use multi-class cross-entropy loss. We also study Wasserstein GAN (WGAN), which is proposed by Arjovsky et al. (2017) to improve the stability of GAN based learning. Its loss function is shown as follows.

$$\mathcal{L}_d = \mathbb{E}_{x \sim X^a}[D(G(x)] - \mathbb{E}_{x \sim X^b}[D(G(x)],$$

Language Families	Languages
Afro-Asiatic	Arabic (ar), Hebrew (he)
Austronesian	Indonesian (id)
IE.Baltic	Latvian (lv)
IE.Germanic	Danish (da), Dutch (nl), English (en), German (de), Norwegian (no), Swedish (sv)
IE.Indic	Hindi (hi)
IE.Latin	Latin (la)
IE.Romance	Catalan (ca), French (fr), Italian (it), Portuguese (pt), Romanian (ro), Spanish (es)
IE.Slavic	Bulgarian (bg), Croatian (hr), Czech (cs), Polish (pl), Russian (ru), Slovak (sk), Slovenian (sl), Ukrainian (uk)
Korean	Korean (ko)
Uralic	Estonian (et), Finnish (fi)

Table 1: The selected 29 languages for experiments from UD v2.2 (Nivre et al., 2018).

here, the annotations are similar to those in the GAN setting.

3 Experiments and Analysis

In this section, we discuss our experiments and analysis on cross-lingual dependency parsing transfer from a variety of perspectives and show the advantages of adversarial training.

Settings. In our experiments, we study single-source parsing transfer, where a parsing model is trained on one source language and directly applied to the target languages. We conduct experiments on the Universal Dependencies (UD) Treebanks (v2.2) (Nivre et al., 2018) using 29 languages, as shown in Table 1. We use the publicly available implementation[4] of the "SelfAtt-Graph" and "RNN-Stack" parsers.[5] Ahmad et al. (2019) show that the "SelfAtt-Graph" parser captures less language-specific information and performs better than the 'RNN-Stack" parser for distant target languages. Therefore, we use the "SelfAtt-Graph" parser in most of our experiments. Besides, the multilingual variant of BERT (mBERT) (Devlin et al., 2019) has shown to perform well in cross-lingual tasks (Wu and Dredze, 2019) and outperform the models trained on multilingual word embeddings by a large margin. Therefore, we consider conducting experiments with both multilingual word embeddings and mBERT. We use aligned multilingual word embeddings (Smith et al., 2017; Bojanowski et al., 2017) with 300 dimensionss or contextualized word representations provided by multilingual BERT[6] (Devlin et al., 2019) with 768 dimensions as the word representations. In addition, we use the Gold universal POS tags to form the input representations.[7] We freeze the word representations during training to avoid the risk of disarranging the multilingual representation alignments.

We select six auxiliary languages[8] (French, Portuguese, Spanish, Russian, German, and Latin) for unsupervised language adaptation via adversarial training. We tune the scaling parameter λ in the range of $[0.1, 0.01, 0.001]$ on the source language validation set and report the test performance with the best value. For gradient reversal (GR) and GAN based adversarial objectives, we use Adam (Kingma and Ba, 2015) to optimize the discriminator parameters, and for WGAN, we use RMSProp (Tieleman and Hinton, 2012). The learning rate is set to 0.001 and 0.00005 for Adam and RMSProp, respectively. We train the parsing models for 400 and 500 epochs with multilingual BERT and multilingual word embeddings respectively. We tune the parameter I (as shown in Algorithm 1) in the range of $[50, 100, 150]$.

Language Test. The goal of training the contextual encoder adversarially with unlabeled data from auxiliary languages is to encourage the encoder to capture more language-agnostic representations and less language-dependent features. To test whether the contextual encoders retain language information after adversarial training, we train a multi-layer Perceptron (MLP) with softmax on top of the *fixed* contextual encoders to perform a 7-way classification task.[9] If a contextual encoder performs better in the language test, it indicates that the encoder retains language specific information.

3.1 Results and Analysis

Table 2 presents the main transfer results of the "SelfAtt-Graph" parser when training on only English (en, baseline), English with French (en-fr), and English with Russian (en-ru). The re-

[4]https://github.com/uclanlp/CrossLingualDepParser

[5]We adopt the same hyper-parameters, experiment settings and evaluation metrics as those in (Ahmad et al., 2019).

[6]https://github.com/huggingface/pytorch-transformers

[7]We concatenate the word and POS representations. In our future work, we will conduct transfer learning for both POS tagging and dependency parsing.

[8]We want to cover languages from different families and with varying distances from the source language (English).

[9]With the source (English) and six auxiliary languages.

Lang	Multilingual Word Embeddings			Multilingual BERT		
	(en)	(en-fr)	(en-ru)	(en)	(en-fr)	(en-ru)
en	**90.23/88.23**	90.01/88.08	89.93/87.93	**93.19/91.21**	92.81/90.97	92.77/90.86
no	80.82/72.94	80.60/72.83	**80.98/73.10**	**85.81/79.03**	85.50/78.64	85.43/78.76
sv	80.33/72.54	79.90/72.16	**80.43/72.68**	85.61/78.34	**85.64/78.58**	85.44/78.33
fr	77.71/72.35	**78.49**†**/73.30**†	78.31/73.29	85.22/80.78	84.76/80.26	**85.91**†**/81.63**†
pt	76.41/67.35	76.88†/67.74	**77.09**†**/67.81**	82.93/73.33	82.71/73.13	**83.43**†**/73.88**†
da	**76.58/68.11**	75.99/67.64	76.25/68.03	82.36/73.53	**82.40**/73.68	82.36/**73.86**†
es	73.76/65.46	**74.14**/65.78	74.08/**65.84**	80.81/72.66	81.11/72.80	**81.38**†**/73.29**†
it	80.89/75.61	**81.33**†**/76.14**†	80.70/75.57	87.07/82.38	86.90/82.22	**87.41/82.67**
hr	62.21/52.67	**63.38**†**/53.83**†	63.11†/53.62†	72.96/62.65	73.39†/62.20	**74.20**†**/63.55**†
ca	73.18/64.53	**73.46**†/64.71	73.40/**64.90**†	80.40/71.42	80.30/71.42	**80.75/71.78**
pl	74.65/62.72	75.65†/63.31†	**75.93/63.60**	81.51/69.25	82.33†/69.91†	**82.48**†**/70.54**†
uk	59.25/51.92	60.58†/**52.72**†	**60.81**†/52.66†	69.98/61.52	70.24/61.61	**71.21**†**/62.84**†
sl	67.51/56.42	68.14/56.52	**68.40/56.87**	75.15/63.12	74.60/62.52	**75.50/63.65**†
nl	68.54/59.99	68.80/60.23	**69.23**†**/60.51**†	76.76/68.35	**76.94**/68.28	76.89/**68.76**†
bg	79.09/67.61	**80.01**†**/68.42**	79.72/68.39	86.82/75.47	87.08/75.40	**87.61**†**/75.94**†
ru	60.91/52.03	61.42†/52.27†	**61.67**†**/52.41**†	71.92/62.09	72.31/62.15	**72.88**†**/62.94**†
de	**71.41/61.97**	70.70/61.41	71.05/61.84	78.66/69.81	78.04/69.23	**79.08**†**/70.26**†
he	55.70/48.08	**57.33**†**/49.37**†	57.15†/49.36†	64.46/**55.82**	64.97†/55.63	**65.30**†/55.76
cs	63.30/54.14	63.94†/54.63†	**64.37**†**/55.08**†	73.78/63.52	**74.57**†/63.86	74.56†/**64.17**†
ro	65.13/53.98	**65.86/54.76**	65.57/54.42	75.10/62.99	75.85†/**63.92**†	**76.06**†/63.78†
sk	66.79/58.23	**67.46**†**/58.77**	67.42†/58.70	76.30/67.38	77.08†/67.57	**77.86**†**/68.28**†
id	49.85/44.09	**52.05**†**/45.76**†	51.57/45.31	56.80/50.24	**57.45**†/50.27	57.30†/**50.70**†
lv	70.45/49.47	70.03/49.38	**70.67**†**/49.61**†	**75.63**/53.93	75.27/53.78	75.62/**54.29**
fi	66.11/48.73	65.84/48.61	**66.28/48.82**	71.59/**53.81**	71.35/53.63	**71.74**/53.79
et	65.01/44.78	65.31†/45.12†	**65.38**†**/45.32**†	71.55/50.98	**71.73/51.27**	71.25/51.16
ar	37.63/27.48	38.72†/**28.00**†	**38.98**†/27.89†	49.27/37.62	50.37†/39.37†	**50.95**†**/39.57**†
la	47.74/34.90	48.80†/35.64†	**49.17**†**/35.73**†	51.83/38.20	51.48/38.00	**52.20/38.28**
ko	**34.44/16.18**	33.98/15.93	34.23/16.08	38.10/20.62	38.03/20.59	**38.98**†**/21.54**†
hi	36.34/27.43	36.72/27.40	**37.37**†**/28.01**†	45.40/35.03	**47.74**†**/35.90**†	46.10†/34.74
Average	65.92/55.86	66.40†/56.22†	**66.53**†**/56.32**†	73.34/62.93	73.55/62.99	**73.88**†**/63.43**†

Table 2: Cross-lingual transfer performances (UAS%/LAS%, excluding punctuation) of the SelfAtt-Graph parser (Ahmad et al., 2019) on the test sets. In column 1, languages are sorted by the word-ordering distance to English. (en-fr) and (en-ru) denotes the source-auxiliary language pairs. '†' indicates that the adversarially trained model results are statistically significantly better (by permutation test, $p < 0.05$) than the model trained only on the source language (en). Results show that the utilization of unlabeled auxiliary language corpora improves cross-lingual transfer performance significantly.

sults demonstrate that the adversarial training with the auxiliary language identification task benefits cross-lingual transfer with a small performance drop on the source language. When multi-lingual embedding is employed, the performance significantly improves, in terms of UAS of 0.48 and 0.61 over the 29 languages when French and Russian are used as the auxiliary language, respectively. When richer multilingual representation technique like mBERT is employed, adversarial training can still improve cross-lingual transfer performances (0.21 and 0.54 UAS over the 29 languages by using French and Russian, respectively).

Next, we apply adversarial training on the "RNN-Stack" parser and show the results in Table 3. Similar to the "SelfAtt-Graph"parser, the "RNN-Stack" parser resulted in significant improvements in cross-lingual transfer from unsupervised language adaptation. We discuss our detailed experimental analysis in the following.

3.1.1 Impact of Adversarial Training

To understand the impact of different adversarial training types and objectives, we apply adversarial training on both word- and sentence-level with gradient reversal (GR), GAN, and WGAN objectives. We provide the average cross-lingual transfer performances in Table 4 for different adversarial training setups. Among the adversarial training objectives, we observe that in most cases, the GAN objective results in better performances than the GR and WGAN objectives. Our finding is in contrast to Adel et al. (2018) where GR was reported to be the better objective. To further investigate, we perform the language test on the encoders trained via these two objectives. We find that the GR-based trained encoders

perform consistently better than the GAN based ones on the language identification task, showing that via GAN-based training, the encoders become more language-agnostic. In a comparison between GAN and WGAN, we notice that GAN-based training consistently performs better.

Comparing word- and sentence-level adversarial training, we observe that predicting language identity at the word-level is slightly more useful for the "SelfAtt-Graph" model, while the sentence-level adversarial training results in better performances for the "RNN-Stack" model. There is no clear dominant strategy.

In addition, we study the effect of using a linear classifier or a multi-layer Perceptron (MLP) as the discriminator and find that the interaction between the encoder and the linear classifier resulted in improvements.[10]

3.1.2 Adversarial v.s. Multi-task Training

In section 3.1.1, we study the effect of learning language-agnostic representation by using auxiliary language with adversarial training. An alternative way to leverage auxiliary language corpora is by encoding language-specific information in the representation via multi-task learning. In the multi-task learning (MTL) setup, the model observes the same amount of data (both labeled and unlabeled) as the adversarially trained (AT) model. The only difference between the MTL and AT models is that in the MTL models, the contextual encoders are encouraged to capture language-dependent features while in the AT models, they are trained to encode language-agnostic features.

The experiment results using multi-task learning in comparison with the adversarial training are presented in Table 5. Interestingly, although the MTL objective sounds contradiction to adversarial learning, it has a positive effect on the cross-lingual parsing, as the representations are learned with certain additional information from new (unlabeled) data. Using MTL, we sometimes observe improvements over the baseline parser, as indicated with the † sign, while the AT models consistently perform better than both the baseline and the MTL model (as shown in Columns 2–5 in Table 5). The comparisons on parsing performances do not reveal whether the contextual encoders learn to encode language-agnostic or dependent features.

Therefore, we perform language test with the MTL and AT (GAN based) encoders, and the results are shown in Table 5, Columns 6–7. The results indicate that the MTL encoders consistently perform better than the AT encoders, which verifies our hypothesis that adversarial training motivates the contextual encoders to encode language-agnostic features.

Lang	(en)	(en-fr)	(en-ru)
en	89.65/87.43	**89.88/87.66**	89.67/87.56
no	80.20/72.11	80.42/72.49	**80.73**†**/72.65**†
sv	81.02/72.95	81.14/**73.44**†	**81.20**/73.37
fr	77.42/72.27	77.45/72.72	**77.78/73.10**
pt	75.94/67.40	76.09/67.47	**76.39**†**/67.85**†
da	76.87/68.06	77.43†/68.62†	**77.92**†**/69.24**†
es	73.92/65.95	74.32†/66.35†	**74.83**†**/66.83**†
it	80.09/75.36	80.98†/76.00†	**81.04**†**/76.06**†
hr	59.53/49.19	60.00†/50.02†	**60.16**†**/50.16**†
ca	73.62/64.97	73.73/65.11	**74.18**†**/65.59**†
pl	71.48/57.43	72.48†/**59.19**†	**72.55**†/58.38†
uk	57.23/49.67	58.38†/**51.04**†	**58.57**†/50.88†
sl	65.48/53.40	66.11†/**54.21**†	**66.23**†/54.09†
nl	67.13/59.15	67.57/59.71†	**67.76**†**/59.96**†
bg	77.28/65.77	77.79†/**66.66**†	**78.02**†/66.53†
ru	58.70/49.34	59.77†/**50.77**†	**59.98**†/50.51†
de	69.71/58.51	70.03/**59.45**†	**70.05**/59.38†
he	52.97/45.73	53.63†/46.49†	**54.72**†**/47.34**†
cs	60.99/51.63	61.60†/52.41†	**61.81**†**/52.45**†
ro	62.01/51.03	62.49/51.30	**63.22**†**/51.91**†
sk	64.44/56.01	65.03†/56.65†	**65.36**†**/56.67**†
id	45.08/40.00	45.46/40.61†	**46.82**†**/41.63**†
lv	70.22/48.46	**71.08**†**/49.10**†	70.76/48.86
fi	65.39/47.78	**65.59/48.31**†	65.42/47.84
et	64.73/43.84	65.01/**44.27**	**65.04**/44.16
ar	30.98/23.83	31.91†/24.72†	**32.83**†**/25.34**†
la	**45.28**/33.08	44.94/32.94	45.12/**33.11**
ko	**33.50/14.36**	32.87/14.10	32.60/14.11
hi	27.63/19.16	**27.66/19.22**	26.72/18.96
Average	64.09/53.93	64.51†/54.52†	**64.74**†**/54.64**†

Table 3: Cross-lingual transfer results (UAS%/LAS%, excluding punctuation) of the RNN-Stack parser on the test sets. '†' indicates that the adversarially trained model results are statistically significantly better (by permutation test, $p < 0.05$) than the model trained only on the source language (en).

3.1.3 Impact of Auxiliary Languages

To analyze the effects of the auxiliary languages in cross-language transfer via adversarial training, we perform experiments by pairing up[11] the source language (English) with six different lan-

[10] This is a known issue in GAN training as the discriminator becomes too strong, it fails to provide useful signals to the generator. In our case, MLP as the discriminator predicts the language labels with higher accuracy and thus fails.

[11] We also conduct experiments on multiple languages as the auxiliary language. For GAN and WGAN-based training, we concatenate the corpora of multiple languages and treat them as one auxiliary language. In these set of experiments, we do not observe any apparent improvements.

AT	SelfAtt-Graph				RNN-Stack			
	en-fr		en-ru		en-fr		en-ru	
	word	sent	word	sent	word	sent	word	sent
GR	66.19	66.21	66.38	66.38	**64.51**	**64.51**	64.52	64.52
GAN	**66.40**	66.29	**66.53**	66.41	64.40	**64.51**	64.63	**64.74**
WGAN	66.24	66.18	66.40	66.27	64.29	64.34	64.57	64.57

Table 4: Average cross-lingual transfer performances (UAS%, excluding punctuation) on the test sets using different adversarial training objective and setting. Multilingual word embeddings are used for these experiments.

Lang (Src. + Aux.)	Auxiliary Language Perf.		Average Cross-lingual Perf.		Lang. Test Perf.	
	AT	MTL	AT	MTL	AT	MTL
en + fr	**78.49/73.30**†	78.26/72.98†	**66.40/56.22**	66.18/56.04	**62.25**	59.94
en + pt	**76.53/67.45**†	75.88/66.75	**66.40/56.22**	66.27/56.08	60.17	**72.02**
en + es	73.66/65.48	**74.04/65.83**†	**66.38/56.24**	66.22/56.12	56.78	**74.52**
en + ru	**61.67/52.41**†	61.08/52.04	**66.53/56.32**	66.35/56.20	37.34	**60.56**
en + de	**71.65/62.11**†	71.17/61.88	**66.41/56.13**	66.18/56.12	61.22	**72.08**
en + la	**49.22/35.94**†	48.04/35.09†	**66.45/56.20**	66.17/56.05	50.04	**64.91**

Table 5: Comparison between adversarial training (AT) and multi-task learning (MTL) of the contextual encoders. Columns 2–5 demonstrate the parsing performances (UAS%/LAS%, excluding punctuation) on the auxiliary languages and average of the 29 languages. Columns 6–7 present accuracy (%) of the language label prediction test. '†' indicates that the performance is higher than the baseline performance (shown in the 2nd column of Table 2).

Aux. lang	Avg. Dist. to other lang	multilingual Word Emb.	multilingual BERT
pt	0.144	66.40/56.22	73.47/63.11
ru	0.146	**66.53/56.32**	73.88/63.43
de	0.151	66.41/56.13	**73.92/63.56**
es	0.151	66.38/56.24	71.71/62.49
fr	0.160	66.40/56.22	73.55/62.99
la	0.242	66.45/56.20	73.69/63.29

Table 6: Average cross-lingual transfer performances (UAS%/LAS%, w/o punctuation) on the test sets using SelfAtt-Graph parser when different languages play the role of the auxiliary language in adversarial training.

guages (spanning Germanic, Romance, Slavic, and Latin language families) as the auxiliary language. The average cross-lingual transfer performances are presented in Table 6 and the results suggest that Russian (ru) and German (de) are better candidates for auxiliary languages.

We then dive deeper into the effects of auxiliary languages trying to understand whether auxiliary languages particularly benefit target languages that are closer to them[12] or from the same family. Intuitively, we would assume when the auxiliary language has a smaller average distance to all the target languages, the cross-lingual transfer performance would be better. However, from the results in Table 6, we do not see such a pattern. For example, Portuguese (pt) has the smallest average distance to other languages among the auxiliary languages we tested, but it is not among the better auxiliary languages.

We further zoom in the cross-lingual transfer improvements for each language families as shown in Table 7. We hypothesis that the auxiliary languages to be more helpful for the target languages in the same family. The experimental results moderately correlate with our expectation. Specifically, the Germanic family benefits the most from employing German (de) as the auxiliary language; similarly Slavic family with Russian (ru) as the auxiliary language (although German as the auxiliary language brings similar improvements). The Romance family is an exception because it benefits the least from using French (fr) as the auxiliary language. This may due to the fact that French is too closed to English, thus is less suitable to be used as an auxiliary language.

4 Related Work

Unsupervised Cross-lingual Parsing. Unsupervised cross-lingual transfer for dependency parsing has been studied over the past few years (Agić et al., 2014; Ma and Xia, 2014; Xiao and Guo, 2014; Tiedemann, 2015; Guo et al., 2015; Aufrant et al., 2015; Rasooli and Collins, 2015; Duong et al., 2015; Schlichtkrull and Søgaard, 2017; Ahmad et al., 2019; Rasooli and Collins, 2019; He et al., 2019). Here, "unsupervised transfer" refers to the setting where a parsing model trained only on the source language is directly

[12]The language distances are computed based on word order characteristics as suggested in Ahmad et al. (2019).

Lang	(en,ru) - en	(en,fr) - en	(en,de) - en	(en,la) - en
IE.Slavic Family				
hr	1.24/0.90	0.43/-0.45	1.52/1.02	0.06/-0.13
sl	0.35/0.53	-0.55/-0.60	-0.04/0.14	-0.17/-0.50
uk	1.23/1.32	0.26/0.09	1.54/1.33	-0.29/-0.09
pl	0.97/1.29	0.82/0.66	0.82/0.98	1.03/0.98
bg	0.79/0.47	0.26/-0.07	0.49/0.41	0.01/0.04
ru	0.96/0.85	0.39/0.06	1.07/1.11	0.20/0.34
cs	0.78/0.65	0.79/0.34	0.91/0.81	-0.08/0.05
sk	1.56/0.90	0.78/0.19	1.88/1.04	0.56/0.66
Avg.	**0.98/0.86**	**0.4/0.03**	**1.02/0.86**	**0.17/0.17**
IE.Romance Family				
pt	0.50/0.55	-0.22/-0.20	0.54/0.80	0.49/0.60
fr	0.69/0.85	-0.46/-0.52	0.49/0.16	0.95/0.86
es	0.57/0.63	0.30/0.14	0.45/0.39	0.44/0.51
it	0.34/0.29	-0.17/-0.16	-0.22/-0.17	0.26/0.18
ca	0.35/0.36	-0.10/0.00	0.64/0.70	0.10/0.28
ro	0.96/0.79	0.75/0.93	1.32/1.32	1.62/1.73
Avg.	**0.57/0.58**	**0.02/0.03**	**0.54/0.53**	**0.64/0.69**
IE.Germanic Family				
en	-0.42/-0.35	-0.38/-0.24	-0.35/-0.25	-0.15/-0.20
no	-0.38/-0.27	-0.31/-0.39	-0.41/-0.15	-0.22/-0.24
sv	-0.17/-0.01	0.03/0.24	-0.12/0.35	-0.02/0.18
da	0.00/0.33	0.04/0.15	-0.15/0.08	-0.46/-0.25
nl	0.13/0.41	0.18/-0.07	0.95/0.89	0.57/0.42
de	0.42/0.45	-0.62/-0.58	1.41/1.40	0.25/0.43
Avg.	**-0.07/0.09**	**-0.18/-0.15**	**0.22/0.39**	**0.00/0.06**

Table 7: Average cross-lingual performance difference between the SelfAtt-Graph parser trained on the source (en) and an auxiliary (x) language and the SelfAtt-Graph parser trained only on English (en) language (UAS%/LAS%, excluding punctuation). We use multilingual BERT in this set of experiments.

transferred to the target languages. In this work, we relax the setting by allowing unlabeled data from one or more auxiliary (helper) languages other than the source language. This setting has been explored in a few prior works. Cohen et al. (2011) learn a generative target language parser with unannotated target data as a linear interpolation of the source language parsers. Täckström et al. (2013) adopt unlabeled target language data and a learning method that can incorporate diverse knowledge sources through ambiguous labeling for transfer parsing. In comparison, we leverage unlabeled auxiliary language data to learn language-agnostic contextual representations to improve cross-lingual transfer.

Multilingual Representation Learning. The basic of the unsupervised cross-lingual parsing is that we can align the representations of different languages into the same space, at least at the word level. The recent development of bilingual or multilingual word embeddings provide us with such shared representations. We refer the readers to the surveys of Ruder et al. (2017) and Glavaš et al. (2019) for details. The main idea is that we can train a model on top of the source language embeddings which are aligned to the same space as the target language embeddings and thus all the model parameters can be directly shared across languages. During transfer to a target language, we simply replace the source language embeddings with the target language embeddings. This idea is further extended to learn multilingual contextualized word representations, for example, multilingual BERT (Devlin et al., 2019), have been shown very effective for many cross-lingual transfer tasks (Wu and Dredze, 2019). In this work, we show that further improvements can be achieved by adaptating the contextual encoders via unlabeled auxiliary languages even when the encoders are trained on top of multilingual BERT.

Adversarial Training. The concept of adversarial training via Generative Adversarial Networks (GANs) (Goodfellow et al., 2014; Szegedy et al., 2014; Goodfellow et al., 2015) was initially introduced in computer vision for image classification and received enormous success in improving model's robustness on input images with perturbations. Later many variants of GANs (Arjovsky et al., 2017; Gulrajani et al., 2017) were proposed to improve its' training stability. In NLP, adversarial training was first utilized for domain adaptation (Ganin et al., 2016). Since then adversarial training has started to receive an increasing interest in the NLP community and applied to many NLP applications including part-of-speech (POS) tagging (Gui et al., 2017; Yasunaga et al., 2018), dependency parsing (Sato et al., 2017), relation extraction (Wu et al., 2017), text classification (Miyato et al., 2017; Liu et al., 2017; Chen and Cardie, 2018), dialogue generation (Li et al., 2017).

In the context of cross-lingual NLP tasks, many recent works adopted adversarial training, such as in sequence tagging (Adel et al., 2018), text classification (Xu and Yang, 2017; Chen et al., 2018), word embedding induction (Zhang et al., 2017; Lample et al., 2018), relation classification (Zou et al., 2018), opinion mining (Wang and Pan, 2018), and question-question similarity reranking (Joty et al., 2017). However, existing approaches only consider using the *target* language as the auxiliary language. It is unclear whether the language invariant representations learned by previously proposed methods can perform well on a

wide variety of *unseen* languages. To the best of our knowledge, we are the first to study the effects of language-agnostic representations on a broad spectrum of languages.

5 Conclusion

In this paper, we study learning language invariant contextual encoders for cross-lingual transfer. Specifically, we leverage unlabeled sentences from auxiliary languages and adversarial training to induce language-agnostic encoders to improve the performances of the cross-lingual dependency parsing. Experiments and analysis using English as the source language and six foreign languages as the auxiliary languages not only show improvements on cross-lingual dependency parsing, but also demonstrates that contextual encoders successfully learns not to capture language-dependent features through adversarial training. In the future, we plan to investigate the effectiveness of adversarial training for multi-source transfer to parsing and other cross-lingual NLP applications.

Acknowledgments

We thank the anonymous reviewers for their helpful feedback. This work was supported in part by National Science Foundation Grant IIS-1760523.

References

Heike Adel, Anton Bryl, David Weiss, and Aliaksei Severyn. 2018. Adversarial neural networks for cross-lingual sequence tagging. *arXiv preprint arXiv:1808.04736*.

Željko Agić, Jörg Tiedemann, Kaja Dobrovoljc, Simon Krek, Danijela Merkler, and Sara Može. 2014. Cross-lingual dependency parsing of related languages with rich morphosyntactic tagsets. In *EMNLP 2014 Workshop on Language Technology for Closely Related Languages and Language Variants*.

Wasi Uddin Ahmad, Zhisong Zhang, Zuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223. PMLR.

Lauriane Aufrant, Guillaume Wisniewski, and François Yvon. 2015. Zero-resource dependency parsing: Boosting delexicalized cross-lingual transfer with linguistic knowledge. In *COLING 2016, the 26th International Conference on Computational Linguistics*, pages 119–130. The COLING 2016 Organizing Committee.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Xilun Chen and Claire Cardie. 2018. Multinomial adversarial networks for multi-domain text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1226–1240.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 50–61, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Timothy Dozat and Christopher D Manning. 2017. Deep biaffine attention for neural dependency parsing. *Internation Conference on Learning Representations*.

Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. Cross-lingual transfer for unsupervised dependency parsing without parallel data. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 113–122.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Goran Glavaš, Robert Litschko, Sebastian Ruder, and Ivan Vulić. 2019. How to (properly) evaluate cross-lingual word embeddings: On strong baselines, comparative analyses, and some misconceptions. In

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 710–721.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Internation Conference on Learning Representations*.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2411–2420.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1234–1244.

Junxian He, Zhisong Zhang, Taylor Berg-Kiripatrick, and Graham Neubig. 2019. Cross-lingual syntactic transfer through unsupervised adaptation of invertible projections. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3211–3223.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language learning with adversarial neural networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 226–237.

Joo-Kyung Kim, Young-Bum Kim, Ruhi Sarikaya, and Eric Fosler-Lussier. 2017. Cross-lingual transfer learning for pos tagging without cross-lingual resources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2832–2838.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Gourab Kundu, Avi Sil, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual coreference resolution and its application to entity linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 395–400. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, Hervé Jégou, et al. 2018. Word translation without parallel data. In *Internation Conference on Learning Representations*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169, Copenhagen, Denmark. Association for Computational Linguistics.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada. Association for Computational Linguistics.

Xuezhe Ma, Zecong Hu, Jingzhou Liu, Nanyun Peng, Graham Neubig, and Eduard Hovy. 2018. Stack-pointer networks for dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1337–1348.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *Internation Conference on Learning Representations*.

Joakim Nivre, Mitchell Abrams, Željko Agić, and et al. 2018. Universal dependencies 2.2. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 328–338.

Mohammad Sadegh Rasooli and Michael Collins. 2019. Low-resource syntactic transfer with unsupervised source reordering. *Proceedings of the 2019*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

Sebastian Ruder, Anders Søgaard, and Ivan Vulic. 2017. A survey of cross-lingual embedding models. *arXiv preprint arXiv:1706.04902*.

Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. 2017. Adversarial training for cross-domain universal dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79.

Michael Schlichtkrull and Anders Søgaard. 2017. Cross-lingual dependency parsing with late decoding for truly low-resource languages. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 220–229. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468. Association for Computational Linguistics.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Samuel L Smith, David HP Turban, Steven Hamblin, and Nils Y Hammerla. 2017. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *Internation Conference on Learning Representations*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Internation Conference on Learning Representations*.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1061–1071. Association for Computational Linguistics.

Jörg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted pos labels. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 340–349.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Wenya Wang and Sinno Jialin Pan. 2018. Transition-based adversarial network for cross-lingual aspect extraction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4475–4481. International Joint Conferences on Artificial Intelligence Organization.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1778–1783.

Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 119–129.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379. Association for Computational Linguistics.

Ruochen Xu and Yiming Yang. 2017. Cross-lingual distillation for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Michihiro Yasunaga, Jungo Kasai, and Dragomir Radev. 2018. Robust multilingual part-of-speech tagging via adversarial training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 976–986, New Orleans, Louisiana. Association for Computational Linguistics.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970.

Bowei Zou, Zengzhuang Xu, Yu Hong, and Guodong Zhou. 2018. Adversarial feature adaptation for cross-lingual relation classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 437–448.

A Dual-Attention Hierarchical Recurrent Neural Network for Dialogue Act Classification

Ruizhe Li[♠], Chenghua Lin[♡], Matthew Collinson[♠], Xiao Li[♠] and Guanyi Chen[♣]
[♠]Department of Computing Science, University of Aberdeen, UK
{r02rl17, matthew.collinson, x.li}@abdn.ac.uk
[♡]Department of Computer Science, University of Sheffield, UK
c.lin@sheffield.ac.uk
[♣]Department of Information and Computing Sciences, Utrecht University, The Netherlands
g.chen@uu.nl

Abstract

Recognising dialogue acts (DA) is important for many natural language processing tasks such as dialogue generation and intention recognition. In this paper, we propose a dual-attention hierarchical recurrent neural network for DA classification. Our model is partially inspired by the observation that conversational utterances are normally associated with both a DA and a topic, where the former captures the social act and the latter describes the subject matter. However, such a dependency between DAs and topics has not been utilised by most existing systems for DA classification. With a novel dual task-specific attention mechanism, our model is able, for utterances, to capture information about both DAs and topics, as well as information about the interactions between them. Experimental results show that by modelling topic as an auxiliary task, our model can significantly improve DA classification, yielding better or comparable performance to the state-of-the-art method on three public datasets.

1 Introduction

Dialogue Acts (DA) are semantic labels of utterances, which are crucial to understanding communication: much of a speaker's intent is expressed, explicitly or implicitly, via social actions (e.g., questions or requests) associated with utterances (Searle, 1969). Recognising DA labels is important for many natural language processing tasks. For instance, in dialogue systems, knowing the DA label of an utterance supports its interpretation as well as the generation of an appropriate response (Searle, 1969; Chen et al., 2018). In the security domain, being able to detect intention in conversational texts can effectively support the recognition of sensitive information exchanged in emails or other communication channels, which is critical to timely security intervention (Verma et al., 2012).

A wide range of techniques have been investigated for DA classification. Early works on DA classification are mostly based on general machine learning techniques, framing the problem either as multi-class classification (e.g., using SVMs (Liu, 2006) and dynamic Bayesian networks (Dielmann and Renals, 2008)) or a structured prediction task (e.g., using Conditional Random Fields (Kim et al., 2010; Chen et al., 2018; Raheja and Tetreault, 2019, CRF)). Recent studies to the problem of DA classification have seen an increasing uptake of deep learning techniques, where promising results have been obtained. Deep learning approaches typically model the dependency between adjacent utterances (Ji et al., 2016; Lee and Dernoncourt, 2016). Some researchers further account for dependencies among both consecutive utterances and consecutive DAs, i.e., both are considered factors that influence natural dialogue (Kumar et al., 2018; Chen et al., 2018). There is also work exploring different deep learning architectures (e.g., hierarchical CNN or RNN/LSTM) for incorporating context information for DA classification (Liu et al., 2017).

It has been observed that conversational utterances are normally associated with both a DA and a topic, where the former captures the social act (e.g., promising) and the latter describes the subject matter (Wallace et al., 2013). It is also recognised that the types of DA associated with a conversation are likely to be influenced by the topic of the conversation (Searle, 1969; Wallace et al., 2013). For instance, conversations relating to topics about *customer service* might be more frequently associated with DAs of type Wh-question (e.g., *Why my mobile is not working?*) and a complaining statement (Bhuiyan et al., 2018); whereas meetings covering administrative

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 383–392
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

topics about resource allocation are likely to exhibit significantly more defending statements and floor grabbers (e.g., *Well I mean - is the handheld really any better?*) (Wrede and Shriberg, 2003). However, such a reasonable source of information, surprisingly, has not been explored in the deep learning literature for DA classification. We assume that modelling the topics of utterances as additional contextual information may effectively support DA classification.

In this paper, we propose a dual-attention hierarchical recurrent neural network with a CRF (DAH-CRF) for DA classification. Our model is able to account for rich context information with the developed dual-attention mechanism, which, in addition to accounting for the dependencies between utterances, can further capture, for utterances, information about both topics and DAs. Topic is a useful source of context information which has not previously been explored in existing deep learning models for DA classification. Second, compared to the flat structure employed by existing models (Khanpour et al., 2016; Ji et al., 2016), our hierarchical recurrent neural network can represent the input at the character, word, utterance, and conversation levels, preserving the natural hierarchical structure of a conversation. To capture the topic information of conversations, we propose a simple automatic utterance-level topic labelling mechanism based on LDA (Blei et al., 2003), which avoids expensive human annotation and improves the generalisability of our model.

We evaluate our model against several strong baselines (Wallace et al., 2013; Ji et al., 2016; Kumar et al., 2018; Chen et al., 2018; Raheja and Tetreault, 2019) on the task of DA classification. Extensive experiments conducted on three public datasets (i.e., Switchboard Dialog Act Corpus (SWDA), DailyDialog (DyDA), and the Meeting Recorder Dialogue Act corpus (MRDA)) show that by modelling the topic information of utterances as an auxiliary task, our model can significantly improve DA classification for all datasets compared to a base model without modelling topic information. Our model also yields better or comparable performance to state-of-the-art deep learning method (Raheja and Tetreault, 2019) in classification accuracy.

To summarise, the contributions of our paper are three-fold: (1) we propose to leverage topic information of utterances, a useful source of contextual information which has not previously been explored in existing deep learning models for DA classification; (2) we propose a dual-attention hierarchical recurrent neural network with a CRF which respects the natural hierarchical structure of a conversation, and is able to incorporate rich context information for DA classification, achieving better or comparable performance to the state-of-the-art; (3) we develop a simple topic labelling mechanism, showing that using the automatically acquired topic information for utterances can effectively improve DA classification.

2 Related Work

Broadly speaking, methods for DA classification can be divided into two categories: multi-class classification (e.g., SVMs (Liu, 2006) and dynamic Bayesian networks (Dielmann and Renals, 2008)) and structured prediction tasks including HMM (Stolcke et al., 2000) and CRF (Kim et al., 2010). Recently, deep learning has been widely applied in many NLP tasks, including DA classification. Kalchbrenner and Blunsom (2013) proposed to model a DA sequence with a RNN where sentence representations were constructed by means of a convolutional neural network (CNN). Lee and Dernoncourt (2016) tackled DA classification with a model built upon RNNs and CNNs. Specifically, their model can leverage the information of preceding texts, which can effectively help improve the DA classification accuracy. A latent variable recurrent neural network was developed for jointly modelling sequences of words and discourse relations between adjacent sentences (Ji et al., 2016). In their work, the shallow discourse structure is represented as a latent variable and the contextual information from preceding utterances are modelled with a RNN.

Kumar et al. (2018) proposed a hierarchical Bi-LSTM model with a CRF for DA classification, where the inter-utterance and intra-utterance information are encoded by a hierarchical Bi-LSTM and the dependency between DA labels is captured by a CRF. Chen et al. (2018) developed a CRF-Attentive Structured Network (CRF-ASN) for DA classification. They applied structured attention network to the CRF layer in order to model contextual utterances and corresponding DAs together. Raheja and Tetreault (2019) achieved the state-of-the-art performance on the SWDA dataset by employing a self-attention mechanism, a CRF

layer and character-level embeddings.

In addition to modelling dependency between utterances, various contexts have also been explored for improving DA classification or joint modelling DA under multi-task learning. For instance, Wallace et al. (2013) proposed a generative joint sequential model to classify both DA and topics of patient-doctor conversations. Their model is similar to the factorial LDA model (Paul and Dredze, 2012), which generalises LDA to assign each token a K-dimensional vector of latent variables. We would like to emphasise that the model of Wallace et al. (2013), only assumed that each utterance is generated conditioned on the previous and current topic/DA pairs. In contrast, our model is able to model the dependencies of all preceding utterances of a conversation, and hence can better capture the effect between DAs and topics.

3 Methodology

Given a training corpus $\mathcal{D} = \langle (C_n, Y_n, Z_n) \rangle_{n=1}^{N}$, where $C_n = \langle u_t^n \rangle_{t=1}^{T}$ is a conversation containing a sequence of T utterances, $Y_n = \langle y_t^n \rangle_{t=1}^{T}$ and $Z_n = \langle z_t^n \rangle_{t=1}^{T}$ are the corresponding labels of DA and topics for C_n, respectively. Each utterance $u_t = \langle w_t^i \rangle_{i=1}^{K}$ of C_n is a sequence of K words. Our goal is to learn a model from $\mathcal{D}$, such that, given an unseen conversation C_u, the model can predict the DA labels of the utterances of C_u.

Figure 1 gives an overview of the proposed Dual-Attention Hierarchical recurrent neural network with a CRF (DAH-CRF). A shared utterance encoder encodes each word w_t^i of an utterance u_t into a vector $\mathbf{h}_t^i$. The DA attention and topic attention mechanisms capture DA and topic information as well as the interactions between them. The outputs of the dual-attention are then encoded in the conversation-level sequence taggers (i.e., $\mathbf{g}_t$ and $\mathbf{s}_t$), based on the corresponding utterance representations (i.e., $\mathbf{l}_t$ and $\mathbf{v}_t$). Finally, the target labels (i.e., y_t and z_t) are predicted in the CRF layer.

3.1 Shared Utterance Encoder

In our model, we adopt a shared utterance encoder to encode the input utterances. Such a design is based on the rationale that the shared encoder can transfer parameters between two tasks and reduce the risk of overfitting (Ruder, 2017). Specifically, the shared utterance encoder is implemented using the bidirectional gated recurrent unit (Cho et al., 2014, BiGRU), which encodes each utterance $u_t = \langle w_t^i \rangle_{i=1}^{K}$ of a conversation C_n as a series of hidden states $\langle \mathbf{h}_t^i \rangle_{i=1}^{K}$. Here, i indicates the timestamp of a sequence, and we define $\mathbf{h}_t^i$ as follows

$$\mathbf{h}_t^i = \overrightarrow{\mathbf{h}}_t^i \oplus \overleftarrow{\mathbf{h}}_t^i \tag{1}$$

where $\oplus$ is an operation for concatenating two vectors, and $\overrightarrow{\mathbf{h}}_t^i$ and $\overleftarrow{\mathbf{h}}_t^i$ are the i-th hidden state of the forward gated recurrent unit (Cho et al., 2014, GRU) and backward GRU for w_t^i, respectively. Formally, the forward GRU $\overrightarrow{\mathbf{h}}_t^i$ is computed as follows

$$\overrightarrow{\mathbf{h}}_t^i = \mathrm{GRU}(\overrightarrow{\mathbf{h}}_t^{i-1}, \mathbf{e}_t^i) \tag{2}$$

where $\mathbf{e}_t^i$ is the concatenation of the word embedding and the character embedding of word w_t^i. Finally, the backward GRU encodes u_t from the reverse direction (i.e. $w_t^K \rightarrow w_t^1$) and generates $\langle \overleftarrow{\mathbf{h}}_t^i \rangle_{i=1}^{K}$ following the same formulation as the forward GRU.

3.2 Task-specific Attention

Recall that one of the key challenges of our model is to capture for each utterance, information about both DAs and topics, as well as information about the interactions between them. We address this challenge by incorporating into our model a novel task-specific dual-attention mechanism, which accounts for both DA and topic information extracted from utterances. In addition, DAs and topics are semantically relevant to different words in an utterance. With the proposed attention mechanism, our model can also assign different weights to the words of an utterance by learning the degree of importance of the words to the DA or topic labelling task, i.e., promoting the words which are important to the task and reducing the noise introduced by less important words.

For each utterance u_t, the DA attention calculates a weight vector $\langle \alpha_t^i \rangle_{i=1}^{K}$ for $\langle \mathbf{h}_t^i \rangle_{i=1}^{K}$, the hidden states of u_t. u_t can then be represented as an attention vector $\mathbf{l}_t$ computed as follows

$$\mathbf{l}_t = \sum_{i=1}^{K} \alpha_t^i \mathbf{h}_t^i \tag{3}$$

In contrast to the traditional attention mechanism (Bahdanau et al., 2015), which only depends on one set of hidden vectors from the Seq2Seq decoder, the DA attention of our model relies on two sets of hidden vectors, i.e., $\mathbf{g}_{t-1}$ of

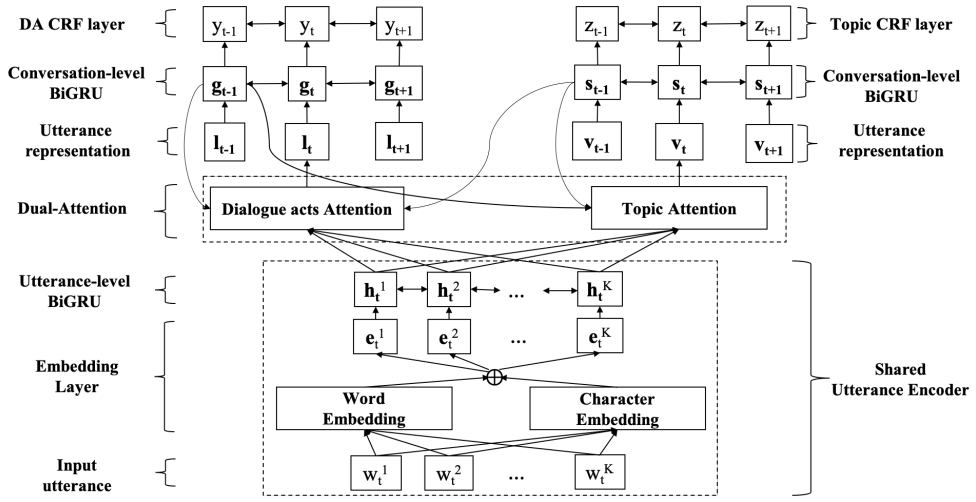


Figure 1: Overview of the dual-attention hierarchical recurrent neural network with a CRF.

the conversation-level DA tagger and $\mathbf{s}_{t-1}$ of the conversation-level topic tagger, where dual attention mechanism can capture, for utterances, information about both DAs and topics as well as the interaction between them. Specifically, the weights $\langle \alpha_t^i \rangle_{i=1}^K$ for the DA attention are calculated as follows:

$$\alpha_t^i = \text{softmax}(o_t^i) \tag{4}$$

$$o_t^i = \mathbf{w}_a^\top \tanh\left(\mathbf{W}^{(\text{act})}(\mathbf{s}_{t-1} \oplus \mathbf{g}_{t-1} \oplus \mathbf{h}_t^i) + \mathbf{b}^{(\text{act})}\right) \tag{5}$$

The topic attention layer has a similar architecture to the DA attention layer, which takes as input both $\mathbf{s}_{t-1}$ and $\mathbf{g}_{t-1}$. The weight vector $\langle \beta_t^i \rangle_{i=1}^K$ for the topic attention output $\mathbf{v}_t$ can be calculated similar to Eq. 3 and Eq. 4. Note that $\mathbf{w}_a$, $\mathbf{W}^{(\text{act})}$, and $\mathbf{b}^{(\text{act})}$ are vectors of parameters that need to be learned during training.

3.3 Conversational Sequence Tagger

CRF sequence tagger for DA. The conversational CRF sequence tagger for DA predicts the next DA y_t conditioned on the conversational hidden state $\mathbf{g}_t$ and adjacent DAs (c.f. Figure 1). Formally, this conditional probability of the whole conversation can be formulated as

$$p\left(y_{1:T}|C;\theta\right) = \frac{\prod_{t=1}^T \Psi\left(y_{t-1}, y_t, \mathbf{g}_t;\theta\right)}{\sum_Y \prod_{t=1}^T \Psi\left(y_{t-1}, y_t, \mathbf{g}_t;\theta\right)} \tag{6}$$

$$\begin{aligned}\Psi\left(y_{t-1}, y_t, \mathbf{g}_t;\theta\right) &= \Psi_{emi}\left(y_t, \mathbf{g}_t\right)\Psi_{tran}\left(y_{t-1}, y_t\right) \\ &= \mathbf{g}_t\left[y_t\right]\mathbf{P}_{y_t, y_{t-1}}\end{aligned} \tag{7}$$

Here the feature function $\Psi(\cdot)$ includes two score potentials: emission and transition. The emission potential Ψ_{emi} regards utterance representation $\mathbf{g}_t$ as the unary feature. The transition potential Ψ_{tran} is a pairwise feature constructed from a $T \times T$ state transition matrix $\mathbf{P}$, where T is the number of DA classes, and $\mathbf{P}_{y_t, y_{t-1}}$ is the probability of transiting from state y_{t-1} to y_t. $C = \langle u_t \rangle_{t=1}^T$ is the sequence of all utterances seen so far, θ is the parameters of the CRF layer. $\mathbf{g}_t$ is calculated in a BiGRU similar to Eq. 1 and Eq. 2:

$$\mathbf{g}_t = \overrightarrow{\mathbf{g}}_t \oplus \overleftarrow{\mathbf{g}}_t \tag{8}$$

$$\overrightarrow{\mathbf{g}}_t = \text{GRU}(\overrightarrow{\mathbf{g}}_{t-1}, \mathbf{l}_t) \tag{9}$$

CRF sequence tagger for topic. The conversational CRF sequence tagger for topic is designed to predict topic z_t conditioned on $\mathbf{v}_t$ and adjacent topics, which can be calculated similar to the formulation of the CRF tagger for DA.

Training the model. Let Θ be all the model parameters that need to be estimated for DAH-CRF. Θ then is estimated based on $\mathcal{D} = \langle (C_n, Y_n, Z_n) \rangle_{n=1}^N$ (i.e., a corpus with N conversations) by maximising the following objective function

$$\begin{aligned}\mathcal{L} = \sum_{n=1}^N [&\log\left(p\left(y_{1:T}^n|C_n;\Theta\right)\right) \\ &+\alpha \log\left(p\left(z_{1:T}^n|C_n;\Theta\right)\right)]\end{aligned} \tag{10}$$

The hyper-parameter α controls the contribution of the conversational topic tagger towards the objective function. In our experiments, $\alpha = 0.5$ is determined using the validation datasets. During

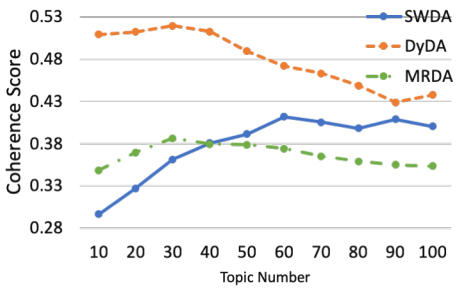


Figure 2: Coherence score of LDA on three datasets.

the test, the optimal DA or topic sequence is calculated using the Viterbi algorithm (Viterbi, 1967).

$$Y' = \arg \max_{y_{1:T} \in Y} p(y_{1:T}|C, \Theta) \quad (11)$$

3.4 Automatically Acquiring Topic Labels

To avoid expensive human annotation and to improve the generalisability of our model, we propose to label the topic of each utterance of the datasets using LDA (Blei et al., 2003). While perplexity has been widely used for model selection for LDA (Lin, 2011; He et al., 2012), we employ a topic coherence measure proposed by (Röder et al., 2015) to determine the optimal topic number for each dataset, which combines the indirect cosine measure with the normalised pointwise mutual information (Bouma, 2009, NPMI) and the Boolean sliding window. Empirically, we found the latter yields much better topic clusters than perplexity for supporting DA classification.

We treat each conversation as a document and train topic models using Gensim with topic number settings ranging from 10 to 100 (using an increment step of 10). Gibbs sampling is used to estimate the model posterior and for each model we run 1,000 iterations. For each trained model, we calculate the averaged coherence score of the extracted topics using Gensim[1], an implementation following (Röder et al., 2015). Figure 2 shows the topic coherence score for each topic number setting for all datasets, from which we determine that the optimal topic number setting for SWDA, DyDA, and MRDA are 60, 30, and 30, respectively.

Based on the optimal models (i.e., a trained LDA model using the optimal topic number setting), we assign topic labels to the datasets with two different strategies, i.e., conversation-level labelling ($conv$) and utterance-level labelling (utt).

[1]https://radimrehurek.com/gensim/models/coherencemodel.html

Dataset	$\|C\|$	$\|T\|$	$\|V\|$	Training	Validation	Testing
SWDA	42	66	20K	1003/193K	112/23K	19/5K
DyDA	4	10	22K	11K/92.7K	1K/8.5K	1K/8.2K
MRDA	5	-	15K	51/77.9K	11/15.8K	11/15.5K

Table 1: $|C|$ is the number of DA classes, $|T|$ is the number of manually labelled conversation-level topic classes, $|V|$ is the vocabulary size. Training, Validation and Testing indicate the number of conversations/utterances in the respective splits.

For conversation-level labelling, we assign the topic label with the highest marginal probability to the conversation based on the corresponding per-document topic proportion estimated by LDA. Every utterance of the conversation then shares the same topic label of the conversation. For utterance-level labelling, there is an additional step to perform inference on every utterance based on corresponding optimal model (e.g., for every utterance of SWDA, we do inference using the LDA trained on SWDA with 60 topics), and assign the topic label with the highest marginal probability to the utterance. Therefore, the topic labels of the utterances of the same conversation could be different for utterance-level labelling.

4 Experimental Settings

4.1 Datasets

We evaluate the performance of our model on three public DA datasets with different characteristics, namely, Switchboard Dialog Act Corpus (Jurafsky, 1997, SWDA), Dailydialog (Li et al., 2017, DyDA), and the Meeting Recorder Dialogue Act corpus (Shriberg et al., 2004, MRDA).

SWDA[2] consists of 1,155 two-sided telephone conversations manually labelled with 66 conversation-level topics (e.g., *taxes*, *music*, etc.) and 42 utterance-level DAs (e.g., *statement-opinion*, *statement-non-opinion*, *wh-question*).

DyDA[3] contains 13,118 human-written daily conversations, manually labelled with 10 conversation-level topics (e.g., *tourism*, *politics*, *finance*) as well as four utterance-level DA classes, i.e., *inform*, *question*, *directive* and *commissive*. The former two classes are information transfer acts, while the latter two are action discussion acts.

MRDA[4] contains 75 meeting conversations anno-

[2]https://web.stanford.edu/~jurafsky/ws97/manual.august1.html

[3]http://yanran.li/dailydialog

[4]http://www1.icsi.berkeley.edu/~ees/dadb/

tated with 5 DAs, i.e., Statement (S), Question (Q), Floorgrabber (F), Backchannel (B), and Disruption (D). The average number of utterances per conversation is 1,496. There are no manually annotated topic labels available for this dataset.

4.2 Implementation Details

For all experimental datasets, the top 85% highest frequency words were indexed. For SWDA and MRDA, we split training/validation/testing datasets following (Stolcke et al., 2000; Lee and Dernoncourt, 2016). For DyDA, we used the standard split from the original dataset (Li et al., 2017). The statistics of the experimental datasets are summarised in Table 1. We represented input data with 300-dimensional Glove word embeddings (Pennington et al., 2014) and 50-dimensional character embeddings (Ma and Hovy, 2016). We set the dimension of the hidden layers (i.e., $\mathbf{h}_t^i$, $\mathbf{g}_t$ and $\mathbf{s}_t$) to 256 and applied a dropout layer to both the shared encoder and the sequence tagger at a rate of 0.2. The Adam optimiser (Kingma and Ba, 2015) was used for training with an initial learning rate of 0.001 and a weight decay of 0.0001. Each utterance in a mini-batch was padded to the maximum length for that batch, and the maximum batch-size allowed was 50.

4.3 Baselines

We compare the proposed DAH-CRF model incorporating utterance-level topic labels extracted by LDA (denoted as DAH-CRF+LDA$_{utt}$) against five strong baselines and two variants of our own models:

JAS[5]: A generative joint, additive, sequential model of topics and speech acts in patient-doctor communication (Wallace et al., 2013);

DRLM-Cond[6]: A latent variable recurrent neural network for DA classification (Ji et al., 2016);

Bi-LSTM-CRF[7]: A hierarchical Bi-LSTM with a CRF to classify DAs (Kumar et al., 2018);

CRF-ASN: An attentive structured network with a CRF for DA classification (Chen et al., 2018);

SelfAtt-CRF: A hierarchical Bi-GRU with self-attention and CRF (Raheja and Tetreault, 2019);

DAH-CRF+MANUAL$_{conv}$: Use the manually annotated conversation-level topic labels (i.e., each utterance of the conversation shares the same topic) for DAH-CRF model training rather than the topic labels automatically acquired from LDA;

DAH-CRF+LDA$_{conv}$: Use conversation-level topic labels automatically acquired from LDA for DAH-CRF model training.

Note that only JAS (a non-deep-learning model) has attempted to model both DAs and topics, whereas all the deep learning baselines do not model topic information as a source of context for DA classification. All the baselines mentioned above use the same test dataset as our models for all experimental datasets.

[5] https://github.com/bwallace/JAS

[6] https://github.com/jiyfeng/drlm

[7] https://github.com/YanWenqiang/HBLSTM-CRF

	Model	SWDA	MRDA	DyDA
Baselines	JAS	71.2	81.3	75.9
	DRLM-Cond	77.0†	88.4	81.1
	Bi-LSTM-CRF	79.2†	90.9†	83.6
	CRF-ASN	80.8†	91.4†	-
	SelfAtt-CRF	**82.9**†	91.1†	-
Ours	DAH-CRF + MANUAL$_{conv}$	80.9	-	86.5
	DAH-CRF + LDA$_{conv}$	80.7	91.2	86.4
	DAH-CRF + LDA$_{utt}$	82.3	**92.2**	**88.1**
	Human Agreement	84.0	-	-

Table 2: DA classification accuracy. † indicates the results which are reported from the prior publications.

5 Experimental Results

5.1 Dialogue Acts Classification

Table 2 shows the DA classification accuracy of our models and the baselines on three experimental datasets. We fine-tuned the model parameters for JAS, DRLM-Cond and Bi-LSTM-CRF in order to make the comparison as fair as possible. The implementation of CRF-ASN and SelfAtt-CRF are not available so we can only report their results for SWDA and MRDA based on the original papers (Chen et al., 2018; Raheja and Tetreault, 2019).

It can be observed that by jointly modelling DA and topics, DAH-CRF+LDA$_{utt}$ outperforms the two best baseline models SelfAtt-CRF and CRF-ASN around 1% on the MRDA dataset. Our model also gives similar performance to SelfAtt-CRF, the baseline which achieved the state-of-the-art performance on the SWDA dataset (i.e., 82.3% vs. 82.9%). While both manually annotated and automatically acquired topic labels are effective, we see that DAH-CRF+LDA$_{utt}$ outperforms both DAH-CRF+MANUAL$_{conv}$ and DAH-CRF+LDA$_{conv}$, i.e., with over 1.6% gain on DyDA and over 1.4% on SWDA (significant; paired t-test $p < .01$). It is also ob-

Model	SWDA	MRDA	DyDA
SAH	76.2	88.5	82.5
SAH-CRF	78.4	89.6	84.1
DAH + LDA$_{utt}$	79.5	91.1	86.0
DAH-CRF + LDA$_{utt}$ (without Dual-Att)	81.0	91.3	86.3
DAH-CRF + LDA$_{utt}$	82.3	92.2	88.1

Table 3: Ablation studies of DA classification.

served that DAH-CRF+MANUAL$_{conv}$ and DAH-CRF+LDA$_{conv}$ perform very similar to each other.

5.2 Ablation Study Results

We conducted ablation studies (see Table 3) in order to evaluate the contribution of the components of our DAH-CRF+LDA$_{utt}$ model, and more importantly, the effectiveness of leveraging topic information for supporting DA classification.

DAH-CRF+LDA$_{utt}$ (without Dual-Att) removes the dual-attention component from DAH-CRF+LDA$_{utt}$, and DAH+LDA$_{utt}$ removes the CRF from DAH-CRF+LDA$_{utt}$ but retaining the dual-attention component. SAH is a Single-Attention Hierarchical RNN model without a CRF, i.e., a simplified version of DAH+LDA$_{utt}$ that only models DAs with topical information omitted. As can be seen in Table 3, DAH+LDA$_{utt}$ achieves over 3% averaged gain on all datasets when compared to SAH, which clearly shows that leveraging topic information can effectively support DA classification. It is also observed that both the dual-attention mechanism and the CRF component are beneficial, but are more effective on the SWDA and DyDA datasets than MRDA.

In summary, while all the analysed model components are beneficial, the biggest gain is obtained by jointly modelling DAs and topics.

5.3 Analysing the Effectiveness of Joint Modelling Dialogue Act and Topic

In this section, we provide detailed analysis on why DAH-CRF+LDA$_{utt}$ can yield better performance than SAH-CRF by jointly modelling DAs and topics. Due to the page limit, our discussion focuses on SWDA and DyDA datasets.

Figure 4 shows the normalized confusion matrix derived from 10 DA classes of SWDA for both SAH-CRF and DAH-CRF+LDA$_{utt}$ models. It can be observed that DAH-CRF+LDA$_{utt}$ yields improvement on recall for many DA classes compared to SAH-CRF, e.g., 23.8% improvement

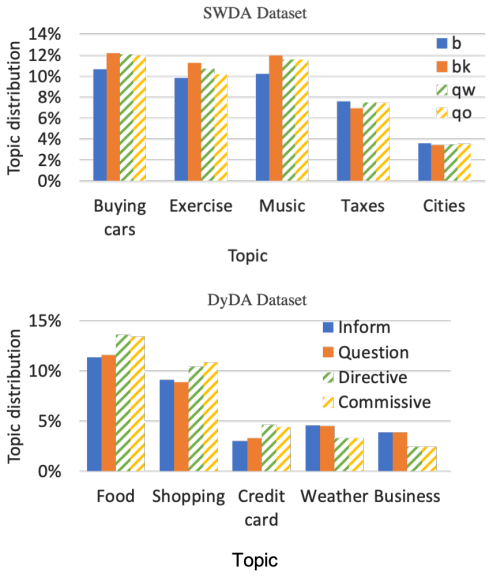


Figure 3: We highlight the prominent topics for some example DAs. The topic distribution of a topic k under a DA label d is calculated by averaging the marginal probability of topic k for all utterances with the DA label d.

on *bk* and 11.7% on *sv*. For *bk* (`Response Acknowledge`) which has the highest improvement level, we see that the improvement largely comes from the reduction of misclassifing *bk* to *b* (`Acknowledge Backchannel`). The key difference between *bk* and *b* is that an utterance labelled with *bk* has to be produced within a question-answer context, whereas *b* is a "continuer" simply representing a response to the speaker (Jurafsky, 1997). It is not surprising that SAH-CRF makes poor prediction on the utterances of these two DAs: they share many syntactic cues, e.g., indicator words such 'okay', 'oh', and 'uh-huh', which can easily confuse the model. When comparing the topic distribution of the utterances under the *bk* and *b* categories (cf. Figure 3), we found topics relating to personal leisure (e.g., buying cars, music, and exercise) are much more prominent in *bk* than *b*. By leveraging the topic information, DAH-CRF+LDA$_{utt}$ can better handle the confusion cases and hence improve the prediction for *bk* significantly.

There are also cases where DAH-CRF+LDA$_{utt}$ performs worse than SAH-CRF. Take the DA pair of *qo* (`Open Question`) and *qw* (`wh-questions`) as an example. *qo* refers to questions like '*How about you?*' and its variations (e.g., '*What do you think?*'), whereas *qw* represents wh-questions which are much

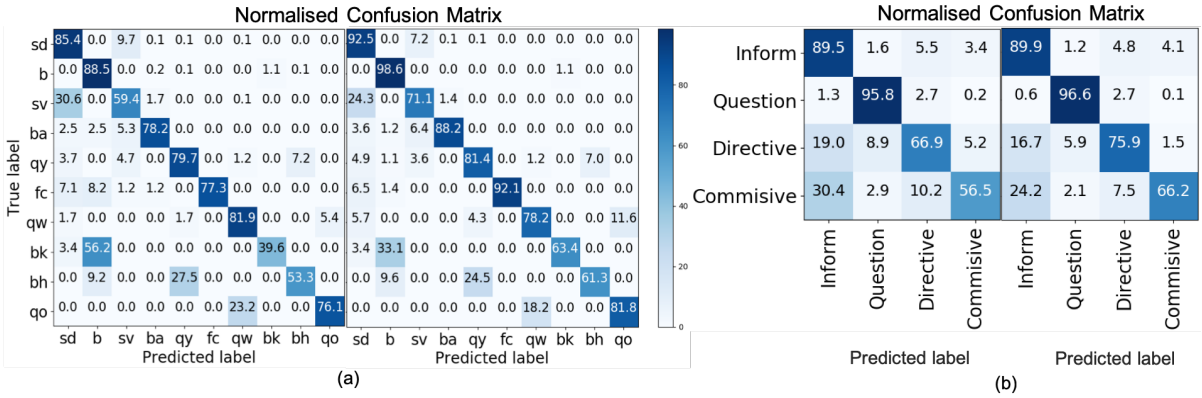


Figure 4: The normalized confusion matrix of DAs using SAH-CRF (left) and DAH-CRF+LDA$_{utt}$ (right) on SWDA (a) and DyDA (b).

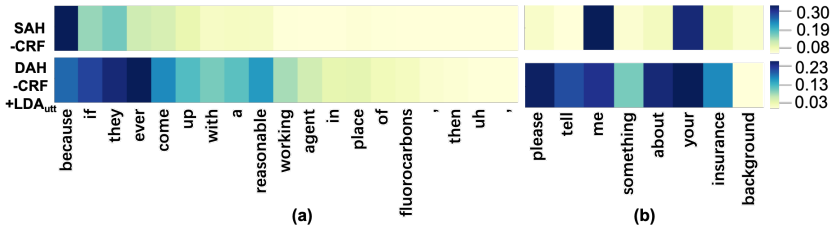


Figure 5: DA Attention visualisation using SAH-CRF and DAH-CRF+LDA$_{utt}$ on (a) SWDA and (b) DyDA datasets. The true labels of the utterances above are *sd* (*statement-non-opinion*) and *Directive*, respectively. SAH-CRF misclassified the DA as *sv* (*statement-opinion*) and *Inform* whereas DAH-CRF+LDA$_{utt}$ gives correct prediction for both cases.

more specific in general (e.g. '*What other long range goals do you have?*'). SAH-CRF gives quite decent performance in distinguishing *qw* and *qo* classes. This is somewhat reasonable, as linguistically the utterances of these two classes are quite different, i.e., the *qw* utterance expresses very specific question and is relatively lengthy, whereas *qo* utterances tends to be very brief. We see that DAH-CRF+LDA$_{utt}$ performs worse than SAH-CRF: a greater number of *qw* utterances are misclassified by DAH-CRF+LDA$_{utt}$ as *qo*. This might be attributed to the fact that topic distributions of *qw* and *qo* are similar to each other (see Figure 3), i.e., incorporating the topic information into DAH-CRF may cause these two DAs to be less distinguishable for the model.

We also conducted a similar analysis on the DyDA dataset. As can be seen from the confusion matrices shown in Figure 4, DAH-CRF+LDA$_{utt}$ gives improvement over SAH-CRF for all the four DA classes of DyDA. In particular, `Directives` and `Commissive` achieve higher improvement margin compared to the other two classes, where the improvement are largely attributed to less number of instances of the `Directives` and `Commissive` classes being mis-classified into `Inform` and `Questions`. Examining the topic distributions in Figure 3 reveals that `Directives` and `Commissive` classes are more relevant to the topics such as *food*, *shopping*, and *credit card*. In contrast, the topics of `Inform` and `Questions` classes are more about *business*, and *weather*.

Finally, Figure 5 shows the DA attention visualisation examples of SAH-CRF and DAH-CRF+LDA$_{utt}$ for an utterance from SWDA and DyDA. For SWDA, it can be seen that SAH-CRF gives very high weight to the word "because" and de-emphasizes other words. However, DAH-CRF+LDA$_{utt}$ can capture more important words (e.g., "if", "reasonable", etc.) and correctly predicts the DA label as *sd*. For DyDA, SAH-CRF only focuses on "me" and "your", but DAH-CRF+LDA$_{utt}$ captures more words relevant to `Directive`, such as "please", "tell", etc. To summarise, DAH-CRF+LDA$_{utt}$ can capture more significant words related to the corresponding DA, by modelling both DAs and topic information with

the dual-attention mechanism.

6 Conclusion

In this paper, we developed a dual-attention hierarchical recurrent neural network with a CRF for DA classification. With the proposed task-specific dual-attention mechanism, our model is able to capture information about both DAs and topics, as well as information about the interactions between them. Moreover, our model is generalised by leveraging an unsupervised model to automatically acquire topic labels. Experimental results based on three public datasets show that modelling utterance-level topic information as an auxiliary task can effectively improve DA classification, and that our model is able to achieve better or comparable performance to the state-of-the-art deep learning methods for DA classification.

We envisage that our idea of modelling topic information for improving DA classification can be adapted to other DNN models, e.g., to encode topic labels into word embeddings and then concatenate with the utterance-level or conversation-level hidden vectors of our baselines, e.g. SelfAtt-CRF. It will also be interesting to explicitly take into account speaker's role in the future.

Acknowledgment

This work is supported by the award made by the UK Engineering and Physical Sciences Research Council (Grant number: EP/P011829/1).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mansurul Bhuiyan, Amita Misra, Saurabh Tripathy, Jalal Mahmud, and Rama Akkiraju. 2018. Don't get Lost in Negation: An Effective Negation Handled Dialogue Acts Prediction Algorithm for Twitter Customer Service Conversations. In *Proc. of ICWSM workshop on Chatbots*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

Zheqian Chen, Rongqin Yang, Zhou Zhao, Deng Cai, and Xiaofei He. 2018. Dialogue act recognition via crf-attentive structured network. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 225–234. ACM.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

Alfred Dielmann and Steve Renals. 2008. Recognition of dialogue acts in multiparty meetings using a switching DBN. *IEEE transactions on audio, speech, and language processing*, 16(7):1303–1314.

Yulan He, Chenghua Lin, and Amparo Elizabeth Cano. 2012. Online sentiment and topic dynamics tracking over the streaming data. In *International Confernece on Social Computing*, pages 258–266.

Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A latent variable recurrent neural network for discourse-driven language models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 332–342.

Dan Jurafsky. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual. *Institute of Cognitive Science Technical Report*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 119–126.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. 2016. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2012–2021.

Su Nam Kim, Lawrence Cavedon, and Timothy Baldwin. 2010. Classifying dialogue acts in one-on-one live chats. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 862–871. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Harshit Kumar, Arvind Agarwal, Riddhiman Dasgupta, and Sachindra Joshi. 2018. Dialogue Act Sequence Labeling Using Hierarchical Encoder With CRF. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ji Young Lee and Franck Dernoncourt. 2016. Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 986–995.

Chenghua Lin. 2011. *Probabilistic topic models for sentiment analysis on the Web*. Ph.D. thesis, University of Exeter.

Yang Liu. 2006. Using SVM and error-correcting codes for multiclass dialog act classification in meeting corpus. In *Ninth International Conference on Spoken Language Processing*.

Yang Liu, Kun Han, Zhao Tan, and Yun Lei. 2017. Using Context Information for Dialog Act Classification in DNN Framework. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2170–2178.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.

Michael Paul and Mark Dredze. 2012. Factorial LDA: Sparse multi-dimensional text models. In *Advances in Neural Information Processing Systems*, pages 2582–2590.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Vipul Raheja and Joel Tetreault. 2019. Dialogue act classification with context-aware self-attention. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3727–3733.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408. ACM.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

John R Searle. 1969. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press.

Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey. 2004. The icsi meeting recorder dialog act (mrda) corpus. In *Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue at HLT-NAACL 2004*, pages 97–100.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3):339–373.

Rakesh Verma, Narasimha Shashidhar, and Nabil Hossain. 2012. Detecting phishing emails the natural language way. In *European Symposium on Research in Computer Security*, pages 824–841. Springer.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Byron C Wallace, Thomas A Trikalinos, M Barton Laws, Ira B Wilson, and Eugene Charniak. 2013. A generative joint, additive, sequential model of topics and speech acts in patient-doctor communication. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1765–1775.

Britta Wrede and Elizabeth Shriberg. 2003. Relationship between dialogue acts and hot spots in meetings. In *2003 IEEE Workshop on Automatic Speech Recognition and Understanding (IEEE Cat. No. 03EX721)*, pages 180–185. IEEE.

Mimic and Rephrase: Reflective listening in open-ended dialogue

Justin Dieter
Square Inc. / Stanford University
justindieter@cs.stanford.edu

Tian Wang
Square Inc.
tian@squareup.com

Gabor Angeli
Square Inc.
gabor@squareup.com

Angel X. Chang*
Simon Fraser University
angelx@sfu.ca

Arun Tejasvi Chaganty*
Square Inc.
arun@squareup.com

Abstract

Reflective listening—demonstrating that you have heard your conversational partner—is key to effective communication. Expert human communicators often *mimic and rephrase* their conversational partner, e.g., when responding to sentimental stories or to questions they don't know the answer to. We introduce a new task and an associated dataset wherein dialogue agents similarly mimic and rephrase a user's request to communicate sympathy (*I'm sorry to hear that*) or lack of knowledge (*I do not know that*). We study what makes a rephrasal response good against a set of qualitative metrics. We then evaluate three models for generating responses: a syntax-aware rule-based system, a seq2seq LSTM neural models with attention (S2SA), and the same neural model augmented with a copy mechanism (S2SA+C). In a human evaluation, we find that S2SA+C and the rule-based system are comparable and approach human-generated response quality. In addition, experiences with a live deployment of S2SA+C in a customer support setting suggest that this generation task is a practical contribution to real world conversational agents.

1 Introduction

Humans in conversation naturally engage in reflective (or active) listening, where they indicate they have heard and understood their partner by repeating or rephrasing what they have said. This strategy has its roots in Rogerian psychology (Rogers, 1951) as a counseling technique meant to build trust and empathy. Dialog agents benefit from the same strategy to keep conversations pleasant, especially when the agent cannot help.

Reflective listening can be formalized into two aspects: (1) *mimicking and rephrasing* the conversational partner's utterance, and (2) incorporating an expressive speech act (Searle, 1976) appropriate for the utterance. For example, in Table 1 (b) we incorporate the speech act *I don't know* on top of the mimicked utterance *when the swimming pool is open*.

Prompt: *Hmm... I'm curious as to whether the swimming pool is open after 7pm?*
✓ a. I do not know that
✓ b. I don't know when the swimming pool is open.
✓ c. I don't know as to whether the swimming pool is open after 7pm.
✗ d. I don't know if you are curious as to whether the swimming pool is open after 7pm.
✗ e. I don't know Hmm... I'm curious as to whether the swimming pool is open after 7pm?

Table 1: Possible responses to indicate the dialog agent doesn't know the answer to a question (the prompt). The last two responses (d,e) are incorrect while the first three (a,b,c) are all acceptable with varying levels of specificity. The best response (b) is the one that is neither too vague (a) nor too verbose and repetitive (c).

In this paper, we propose a new task of generating mimic rephrasals for a given speech act. Table 1 illustrates the task with an example prompt and possible range of responses. The task is non-trivial to handle as naively putting the two parts together will result in responses that are either ungrammatical (e) or do not select the appropriate clause to rephrase (d). In our example, the first three responses all correctly convey the "I don't know" message. However, the blanket response (a) is overly vague and does not convey any understanding. Response (c) is specific but overly verbose and robotic. The best response is (b) since it contains enough details to signal understanding while remaining concise.

To get this best response, the agent must ig-

*These authors contributed equally.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 393–403
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

nore user flourishes (*"Hmm... I'm curious"*), identify the relevant portions of the prompt, correctly rephrase keywords (replace "I" with "you"), and coordinate arguments (using *"when"*).

By simulating reflective listening, we believe that mimic rephrasals will allow goal-oriented dialog systems to still respond naturally to open-ended input from users. In that vein, there has been renewed interest in open-ended dialog systems using neural models. However, Li et al. (2016a) have noted that naïve neural models tend to generate repetitive and dull responses such as "I don't know". While several attempts have been made to control various aspects of generation and hence produce more diverse output (Li et al., 2018; Hu et al., 2017; Logeswaran et al., 2018), we instead focus on expressing a single speech act (e.g., "I don't know"), but grounding it in diverse open-ended settings to simulate reflective listening.

In this paper, we examine what makes for a *good* response for a given speech act. We create two datasets IDONTKNOW and EMOTIVE focusing on two speech acts demonstrating reflective listening, respectively stating that we do not know an answer and expressing sympathy. We analyze the quality of responses along different dimensions such as fluency (is the response grammatical?), appropriateness (is the response on topic?), specificity, repetitiveness, and conciseness. We also compare responses from rule-based and neural models to gain insight into the strengths/weaknesses of different models at this task. We demonstrate that the rule-based model is repetitive but performs well for simple cases, while a sequence-to-sequence model with attention (Bahdanau et al., 2015) and a copying mechanism (Gu et al., 2016) has more varied responses and compares favorably. We release our dataset, code, and experiments to the community. [1]

2 Task: Mimic Rephrasals

In this section, we introduce the task of *Mimic Rephrasal* more formally. We use the term *speech act* to describe the information we want to convey to our conversational partner. For example, a lack of knowledge, sympathy, etc. We define a *prompt* as an utterance by a conversational partner that should trigger some form of speech act. For example, "where is my car?" could be a prompt for the speech act conveying a lack of knowledge. Note that detection of these prompts and classification into the appropriate speech act—while important for a real-world system—is outside the scope of this task.

The task is as follows: given a prompt and the target speech act, generate a *mimic rephrasal* of that prompt which conveys the speech act. We explore the two use-cases of rephrasing lack of knowledge (IDONTKNOW) and sympathy (EMOTIVE).

The important goal of the task to generate a response that makes the user feel that they have been heard and understood. Directly measuring this is difficult. Instead, we propose five metrics to characterize the quality of mimic rephrasals:

- **appropriateness** Did the response include the topic of interest in the rephrasal?
- **fluency** Is the response grammatical?
- **specificity** How much detail from the input prompt is captured?
- **conciseness** Is the response to the point?
- **repetitiveness** Is the response repetitive?

Looking at Table 1: (d) is not *appropriate* and (e) has low *fluency*. (a) to (c) are both *appropriate* and *fluent* with varying *specificity* (from low to high) and *conciseness* (from high to low). Intuitively, there is a tradeoff between specificity and repetitiveness: it should neither be too vague nor too repetitive.

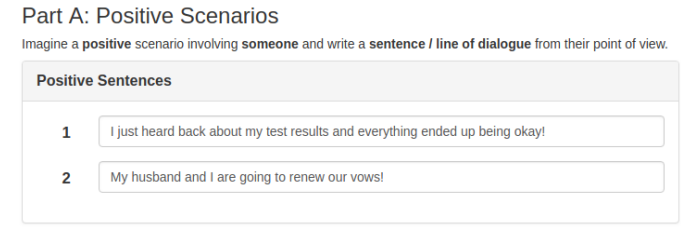
3 Dataset

Section 3.1 describes our process for collecting data, to document our dataset creation and to describe how to collect data for other speech acts—e.g., expressing gratitude, soliciting confirmation of intents, etc. The subsequent section (Section 3.2) describes some statistics of the two datasets in this paper: EMOTIVE and IDONTKNOW.

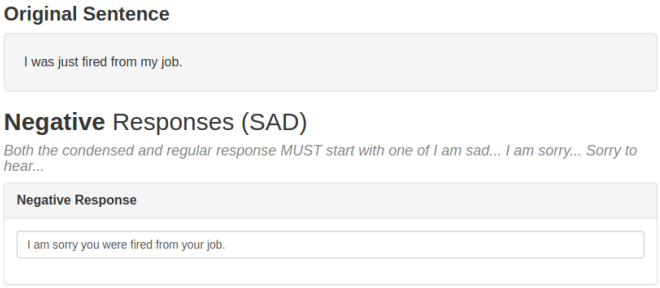
3.1 Data collection

Our data collection pipeline has two phases. In the first phase, workers were asked to come up with a *prompt* or scenario. For example, a question to ask the dialog agent, or a sentiment laden scenario. During this phase, workers were asked to be as creative as possible, to explore a variety of sentence structures and lengths in the way they phrase their prompts, and diversity in the topics covered. In the second phase, we asked another set of workers to generate multiple *responses* each to the prompt.

[1] https://github.com/square/MimicAndRephrase/



(a) Interface when generating prompts



(b) Interface when generating mimic rephrasals

Figure 1: Our data collection pipeline. First, crowdworkers are asked to generate a given *prompt* or scenario (a). Then, a different worker is asked to generate the *mimic rephrasal* of the prompt generated by the first worker (b).

IDontKnow
Prompt: I am legally resident in Northern Ireland, where can I apply for an Irish visa **Mimic Rephrasal**: I do not know where you can apply for an Irish visa
P: When I register a domain, do I receive a website and a web hosting space **MR**: I do not know if you receive a website and a web hosting space when you register a domain
P: I'm having difficulty signing up. Whom can I contact **MR**: I do not know who you can contact about your difficulty signing up
Emotive
P: The sisters were able to reunite after 20 years **MR**: I am happy to hear the sisters were able to reunite after all this time
P: The future looks brighter than I ever imagined **MR**: I 'm happy that your future looks bright to you
P: My phone fell into the toilet and it's ruined now. **MR**: I am sad that your phone is ruined because it fell into the toilet

Table 2: Examples of mimic rephrasals in the IDontKnow and Emotive datasets collected in this paper. Each example has a *prompt* (the utterance from the conversational partner), and a *mimic rephrasal*: the utterance that should be returned by the dialog agent.

We found that splitting the task up into these two steps—generating prompts and then responses—improved the quality of our collected sentence pairs.

The interface used by Mechanical Turk workers is shown in Figure 1. Workers are first asked to generate a number of prompts (scenarios) in Figure 1 (a). Once these prompts are collected, a different set of workers were asked to generate responses to the prompts, completing our dataset (see Figure 1 (b)). Workers were paid \$0.10 per sentence in the prompt generation task, and \$0.07 per sentence in the response generation task.

3.2 Dataset statistics

We use the method described in Section 3.1 to collect two datasets IDontKnow and Emotive. Both datasets are split into train/dev/test splits with a ratio of 70/15/15%.

IDontKnow is a dataset for indicating that we don't know the answer to a question, or cannot execute a request. **Emotive** is a dataset for expressing sympathy for the topic of the prompt, with a balanced distribution of positive and negative sentiment. Examples for the two datasets can be found in Table 2, and statistics are given in Table 3. The modest size of the training set (10 189) means that a good fraction of the test set contains out of vocabulary words: the 1 377 test examples contain 512 words not seen during training, motivating our use of a copy mechanism.

We report statistics both on the full mimic rephrasal (MR), as well as for just the *Mimic* por-

	IDK	EMOTIVE
Dataset size		
Training pairs	6435	3887
Development pairs	1377	834
Test pairs	1377	828
Sentence length		
Prompt mean token count	11.7	11.0
Mimic mean token count	8.8	9.1
MR mean token count	12.9	10.2
Train Vocabulary		
Prompt vocabulary size	5703	4060
MR vocabulary size	5136	3200
Prompt/MR Jaccard Sim	0.83	0.61

Table 3: Statistics about the IDONTKNOW and EMOTIVE datasets. *Mimic* is here taken to be the mimicked utterance, without the preceding "I am happy/sorry/etc." or "I don't know". *MR* is the complete mimic rephrased response.

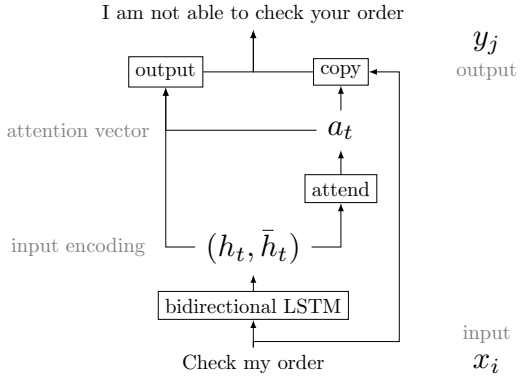


Figure 2: An outline of the sequence to sequence model with attention and a copying mechanism. The input phrase is *"Check my order"* with a correct output of *"I am not able to check your order"*.

tion of the mimic rephrasal. For example, for the MR "I do not know where you can apply for an Irish visa", the *Mimic* portion would be "where you can apply for an Irish visa." While the average MR is slightly longer than the original prompt, the Mimic portion averages 2.9 tokens (25%) shorter than the prompt. In addition, the high Jaccard similarity between the prompt and mimic portion suggests the task involves selecting key portions of the original sentence. [2]

4 Methods

In this section we describe three models for the task described above. These include a rule based baseline constructed with deterministic syntactic transformations as well as trained neural models.

4.1 Rule based baseline

As a naïve baseline, we use a set of hand-written syntactic rephrasing rules using Stanford CoreNLP (Manning et al., 2014). For example, for the IDONTKNOW dataset we developed 8 rules that match a Semgrex (Chambers et al., 2007) pattern to an associated dependency graph manipulation algorithm. For example, the rule based system matches the phrase *"What is the difference between the debt and the deficit?"* to a general type of pattern where the verb (in this case *"is"*) needs to be extracted from the dependency graph and reattached at the end to produce *"I do not know what the difference between the debt and the deficit is."* The EMOTIVE rule based system extracts the root clause from the constituency parse of the sentence and adds enclosing phrasing ("Sorry to hear that..."). Both systems also use simple string manipulation to replace pronouns and correct casing. We note that the rules were developed by iterating on the training data. In the appendix, we include a histogram of how often each rule was fired in the IDONTKNOW rule based system.

Although this is a strong baseline, it has some weaknesses. Writing and maintaining the rule set is difficult and time consuming. The Semgrex patterns and accompanying transformations are non-trivial and requires expert time to develop and maintain. Additionally, it is difficult to deterministically decide which portions of the prompt to keep or drop in the rephrasal. For instance, *"I found out someone has been stealing from me"* should drop the *found out* and respond with: *"Sorry to hear that someone has been stealing from you"*.

4.2 Neural Models

To address these issues, we develop a series of neural models for the task. Formally, let $\mathbf{x} = x_1, \ldots, x_n$ be the source sentence, and $\mathbf{y} = y_1, \ldots, y_m$ be the generated sequence of output tokens. We define a seq2seq model similar to existing neural MT models (Cho et al., 2014) for generating $\mathbf{y}$ given an input $\mathbf{x}$, as well as a model augmented with a copy mechanism.

Input embedding. All models use concatenated ELMo (Peters et al., 2018) and GloVE (Pennington et al., 2014) embedding for the input embeddings. The model architectures used for the EMOTIVE and IDONTKNOW datasets are identical with one

[2] Jaccard similarity is computed as the intersection over union of lemmatized non-stopword tokens between the prompt and Mimic portion.

exception. For the EMOTIVE task, an extra bit is appended to the word embeddings to specify whether the input has a positive or negative sentiment. [3]

Baseline neural model. Our baseline neural model is a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) with attention (Bahdanau et al., 2015). Formally, the *encoder* outputs a set of hidden states for each token given by $\{\bar{h}_1, \ldots, \bar{h}_n\} = LSTM\{m(x_1, \ldots, x_n)\}$ for word embeddings $m(x)$. The decoder is also an LSTM with hidden state initialized to the sum of the final hidden states of the forward and backward LSTMs contained in the bidirectional LSTM encoder. For encoder hidden state $\bar{h}_s$ and decoder hidden state h_t, the attention score $a_t(s)$ is defined as

$$a_t(s) = \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_{s'} \exp(\text{score}(h_t, \bar{h}_{s'}))}$$

where $\text{score}(h_t, \bar{h}_s) = \boldsymbol{v}_a^T \cdot \tanh(W_a[h_t; \bar{h}_s])$ and $\boldsymbol{v}_a$ and W_a are learnable parameters.

Decoding uses a modified beam search (see Section 4.3),[4] and the model is trained on the following cross entropy loss function:

$$J = \sum_{i=1}^{m} -\log p_v(y_i = y_i^* | y_{<i}, x)$$

where y_i^* is the expected output token given in the training data and $\log p_v(y_i = y_i^* | y_{<i}, x)$ is a distribution over the models vocabulary computed from the logits outputted by the model.

Neural model with copying. The most effective neural model we implemented augments the baseline neural model with a copying mechanism (Gu et al., 2016). This allows the model to generalize better to unseen vocabulary, and more strongly enforces the core tenant of the task: that we should be *mimicking* the prompt. An overview of the model is shown in Figure 2.

The key difference from the baseline neural model is that we now generate output tokens using a combined softmax over the model's vocabulary and the tokens in the input:

$$\log p(y_i = y_i^* | y_{<i}, x) = \log p_v(y_i = y_i^* | y_{<i}, x) + \sum_{\{j | x_j = y_i^*\}} \log p_c(y_i = \text{copy}(x_j) | y_{<i}, x),$$

where $\log p_v(\cdot)$ is the same as before, and $\log p_c(\cdot)$ is a distribution over the words in the input. For further details we defer the reader to (Gu et al., 2016). Similar to the baseline model, we decode the model using a modified beam search and train it using a cross-entropy loss.

4.3 Modified Beam Search

We use a modified version of beam search (Huang et al., 2017) when generating output tokens to favor longer responses. The modified beam search first calculates the average ratio of output tokens to input tokens from the dev set, k. We then compute the average logit value of an individual output token, $r(y_i)$, over all outputs produced on the dev set input, r_{avg}. A modified perplexity, $\tilde{sc}(\mathbf{y}, \mathbf{x})$, is used to determine which beams to prune, where $\mathbf{x}$ is a series of input tokens and $\mathbf{y}$ is a proposed series of output tokens (a beam):

$$\tilde{sc}(\mathbf{y}, \mathbf{x}) = sc(\mathbf{y}) + r_{avg} \cdot \min\{\text{len}(\mathbf{y}), k \cdot \text{len}(\mathbf{x})\}$$

where $sc(\mathbf{y}) = \sum_{i=1}^{\text{len}(\mathbf{y})} r(y_i)$ is the standard perplexity for the generated output.

Additionally, we found that for a given output $\bar{\mathbf{y}}$, the score $\tilde{sc}(\bar{\mathbf{y}}, \mathbf{x})$ provides a good measure of generation quality and is useful when filtering out poor or unacceptable output.

4.4 Training

All models were implemented and trained using PyTorch (Paszke et al., 2017). The Adam (Kingma and Ba, 2014) optimizer was used for all gradient based optimization.

We used a randomized hyperparameter grid search to determine the learning rate, number of layers, dropout, and the dimensions of the hidden layers. We used a learning rate of 0.000718 for all optimization. A dropout value of 0.1 is used for all models. All LSTMs are bidirectional with a single layer. Both sequence to sequence models for the IDONTKNOW task use a hidden size of 524 within the LSTM, a hidden size of 100 for the attention layer, a hidden size of 638 for the copy layer, and a dropout value of 0.1. Both sequence to sequence models for the EMOTIVE task use a hidden size of 600 within the LSTM, a hidden size of 200 for the attention layer, a hidden size of 650 for the copy layer, and a dropout value of 0.1.

5 Experiments

We study what makes a good response by correlating human judgments of goodness (based on a

[3] We note that the rule-based system simply used different rules for different settings of this bit.

[4] Initial experiments show that this modified beam search worked better

Model	IDontKnow Dev	IDontKnow Test	Emotive Dev	Emotive Test
BLEU				
Rule-based	78.9	79.4	**47.0**	**46.6**
S2SA	63.3	63.1	32.9	34.2
S2SA+C	**79.9**	**79.7**	44.6	46.3
METEOR				
Rule-based	84.6	85.3	**54.4**	**54.1**
S2SA	74.1	73.6	37.1	38.1
S2SA+C	**88.4**	**88.5**	51.5	52.8

Table 4: BLEU and METEOR scores evaluated on the Dev and Test sets. The S2SA+C performs the best on IDontKnow and the rule-based performs the best on Emotive.

5-point Likert scale) to the set of qualitative metrics we defined in Section 2. The average score is 4.2 for IDontKnow and 3.7 for Emotive. We also evaluated the performance of different models on these datasets using: (1) an automated BLEU and METEOR evaluation, (2) an A/B study comparing the model output with the gold response, and (3) the qualitative metrics. We conclude with a qualitative error analysis and some observations from a live deployment of the neural rephrasal model.

We compare the responses generated by three models: (1) the **Rule-based** baseline; (2) **S2SA**: neural model consisting of a seq2seq model with attention, and (3) **S2SA+C**: neural seq2seq model with attention and copying.

5.1 Results

BLEU/METEOR Following prior work on text generation, we use BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) to compare the performance of our model.[5] From the results shown in Table 4, the neural model with copying (**S2SA+C**) are the rule-based baseline have comparable performance, with both significantly outperforming the baseline neural model (**S2SA**).

A/B Test We perform a human evaluation of our model outputs by creating an A/B test where evaluators specify a preference for either the model output, or the gold human response (see Figure 3). A perfect score on this evaluation would be 50%, indicating that the model and human response are indistinguishable. We ran this test on 305 examples selected from the test set. For each example which was not identical to the gold output, five Turk Workers were asked to choose which response they preferred. Table 5 shows the result of the study, showing that the copy mechanism outperforms both the rule-based baseline and the **S2SA** model for IDontKnow. For Emotive, the rule-based baseline is preferred.

[5]Specifically, we used NLTK (Loper and Bird, 2002) to compute both BLEU and METEOR scores with one human reference for each example. The BLEU score is weighted equally between 1-grams, 2-grams, 3-grams, and 4-grams

Model	IDontKnow P%	IDontKnow (I%)	Emotive P%	Emotive (I%)
Rule-based	38.0	(28.9)	**39.7**	(0.25)
S2SA	16.4	(11.5)	12.1	(0.25)
S2SA+C	**46.7**	(44.3)	35.9	(1.0)

Table 5: The percent of model responses that were (P)referred over the human responses in the A/B test portion of the user study on 305 IDontKnow examples and 400 Emotive examples. When the model's response was identical to the human's, we assume it is preferred 50% of the time: the percentage of these examples is reported in the (I) column.

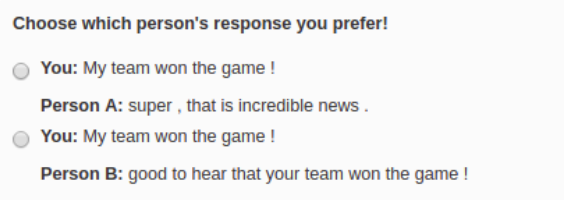


Figure 3: Example question in A/B test. The prompt asks the crowdworker to choose between the human response and the model output. In this example, one would prefer the Person B's response because it is more specific and exhibits reflective listening.

Qualitative Metrics To gain insight into the types of errors the different models are making, we elicited human assessment of three of the metrics defined in Section 2:

1. **Appropriateness** Evaluators make a binary choice as to whether the response included the correct part of the prompt.
2. **Fluency** Evaluators assess the grammatical correctness of the response by selecting on a 3 point Likert scale ranging from "not fluent" to "somewhat fluent" to "fluent". Scores are normalized to 1.
3. **Specificity** We present evaluators with a response and ask them to pick the original prompt from 4 choices (the original prompt, two distractors, and "none/multiple applies"). The distractors are chosen from the nearest neighbors of the prompt using an averaged

Model	App.	Flu.	Spec.	Con.	Rep.
IDONTKNOW					
Human	*93.1*	*91.75*	*74.7*	*1.18*	*65.1*
Rule-based	86.3	85.2	79.3	1.25	74.4
S2SA	49.5	66.8	24.4	1.12	45.7
S2SA+C	**92.4**	**86.0**	**77.5**	**1.22**	**70.4**
EMOTIVE					
Human	*93.2*	*88.5*	*61.3*	*0.97*	*36.3*
Rule-based	**94.4**	**91.7**	90.0	1.29	84.4
S2SA	29.8	74.4	7.0	**0.97**	17.7
S2SA+C	76.4	76.1	**63.5**	1.04	**49.7**

Table 6: Assessment and measures of **App**ropriateness, **Flu**ency, **Spec**ificity, **Con**ciseness and **Rep**etitiveness. We **bold** the highest scores for App. and Flu., and closest to human for Spec., Con., and Rep.

Model	App.	Flu.	Spec.	Con.	Rep.
IDONTKNOW					
Human*	**0.34**	0.14	0.17	**-0.39**	**0.38**
Rules	**0.39**	**0.42**	0.17	-0.03	0.08
S2SA	**0.54**	**0.30**	**0.35**	0.00	0.23
S2SA+C	**0.45**	**0.48**	0.09	-0.19	0.21
EMOTIVE					
Human*	**0.52**	**0.59**	0.05	-0.17	-0.08
Rules	0.12	0.04	-0.08	-0.08	0.01
S2SA	**0.63**	0.23	0.22	-0.03	**0.33**
S2SA+C	**0.41**	**0.41**	0.00	0.02	-0.00

Table 7: Correlations between diagnostic metrics and human quality judgments for responses in the two datasets, with **bold** indicating statistically significant correlations. For all model responses, we use the A/B test preferences as a measure of quality judgment. For the human responses, we use a 5-pt Likert scale of the "goodness" of response as a quality judgment.

GloVE sentence embedding.

We also used the following automatic metrics as proxies for the remaining qualitative metrics:

1. **Conciseness** The length of the response normalized to the length of the prompt (smaller is more concise).
2. **Repetitiveness** We use METEOR (Banerjee and Lavie, 2005) to measure the overlap between prompt and response.

Table 6 shows the results of the human judgment on 400 generated responses for the test set. The **rule-based** model does well on the EMOTIVE dataset. The **S2SA** model is overall worst on most metrics, except conciseness. On the other hand, the **S2SA+C** model performs best on appropriateness and fluency for the IDONTKNOW dataset, and compares favorably with the rule-based model for other metrics. We note that the **S2SA+C** model most closely matches the amount of specificity, conciseness, and repetitiveness in the human responses. Examples of human responses with their corresponding metrics is provided in the appendix, along with additional responses from the models and error analysis.

5.2 Analysis

Next, we look at the correlation of our qualitative metrics to overall human quality judgments. Results of this analysis are in Table 7 and we provide additional visualizations in the appendix. The human response goodness score correlated positively with appropriateness, fluency, and to some extent with repetitiveness. On the other hand it correlated negatively with conciseness (i.e., shorter responses are preferred), while correlation with specificity was less pronounced. This seems to indicate that good responses are characterized by being appropriate and fluent, while having an appropriate level of detail (indicated by some amount of repetitiveness balanced with conciseness). We see a similar trend for the model responses: appropriateness and fluency are the most important attributes for when a model's response is preferred over the human's.

To get a better qualitative understanding of the model's performance, we studied the responses generated by our models (see Table 8 for examples). For simple sentences, the IDONTKNOW **rule-based** responses are reasonable. However, for more complex sentences, it becomes challenging to identify relevant subclauses or to handle non-trivial constructions like conditional clauses (see aquarium example). As a result, the EMOTIVE **rule-based** responses, while grammatical, tend to be overly verbose.

Both **S2SA** and **S2SA+C** are good at producing relatively fluent sentences, and performing the correct pronoun replacements ("you" with "we" and "I"). **S2SA** responses are often off-topic and inappropriate, with the model generating words that related to the topic but prone to drift (e.g., the teapot example). Since we train on a very small dataset, many words in the prompt are not seen during training. While the input word embedding can help during encoding, the decoder is nonetheless unable to generate words it has not seen during training.

The two biggest errors **S2SA+C** makes are incorrectly identifying the relevant parts of the question

	IDONTKNOW	EMOTIVE
Prompt	If I win my case, what am I entitled to	I dropped the jar and it shattered everywhere
Human	I do not know what you are entitled to	I am sorry to hear the jar broke
Rule-based	I do not know what you are entitled to	I am sorry you dropped the jar and it shattered everywhere
S2SA	I do not know what you are required to	i am sorry to hear about the ice
S2SA+C	I do not know what you are entitled to	i am sorry you dropped the jar
Prompt	At what temperature should I heat my water in a Staub teapot	The dentist told me that my insurance did n't cover dental
Human	I do not know what temperature you should heat your water to in a Staub teapot	I am sorry your insurance does n't cover dental
Rule-based	I do not know at what temperature you should heat your water in a Staub teapot	I am sad the dentist told you that your insurance did n't cover dental
S2SA	i am not able to tell you what long it you should handle your tea in a tsunami machine	sorry to hear that you ca n't afford your insurance insurance insurance insurance
S2SA+C	i do not know at what you should heat your water in a staub teapot	sorry to hear that the dentist did n't cover dental
Prompt	Are the animals at your aquarium humanely treated	Did I tell you that I won $ 500 at bingo
Human	I do not know if the animals at our aquarium are humanely treated	I am glad you won it
Rule-based	I do not know of the are animals at my aquarium humanely treated	I'm happy did you tell me that you won $ 500 at bingo
S2SA	i do not know if the animals are at our zoo are allowed	i am happy you won the lottery
S2SA+C	i do not know if the animals are at our aquarium are treated treated	i am happy that you won $ 500 at bingo

Table 8: Example responses from the different models, with errors highlighted in red. Note that the **S2SA** model tend to introduce random terms and **S2SA+C** model will retain numbers such as $ 500.

(e.g., the response *"I do not know what reporter's transcript deposit are"* to the question *"What are Reporter's Transcript Deposit Costs?"*) and grammatical errors when rephrasing (e.g., the response *"I do not know when the free trial is end."* to the question *"When does the free trial end?"*).

5.3 Observations from a Live Deployment

We deployed the **S2SA+C** model as part of a live chatbot for customer service that helped answer customer queries and perform simple tasks like tracking their packages.[6] As context, customers would ask the chatbot questions (e.g., "how do I ship pets?" or "I want to change the delivery address for my package") which were matched against a knowledge base containing frequently asked questions. If a question similarity model was unable to find a match, we tried to communicate to the user that we could not answer their request. Prior to this work, the chatbot would respond with a generic backoff message: "I'm sorry I didn't understand something you said" which resulted in users repeatedly rewording their request even if their request was genuinely outside of the chatbot's knowledge base, and ultimately expressing frustration with the chatbot.

We incorporated mimic rephrasals into our system by responding with the output generated by the S2SA+C model trained on the IDONTKNOW dataset if its score, $\tilde{sc}(\mathbf{y}, \mathbf{x})$, was higher than a fixed threshold, and using the previous backoff response if not. We observed that when the model replied with a mimic rephrasal, users usually responded with gratitude, e.g. "Thanks for letting me know!", and either continued by asking a different question or leaving the conversation. Presented with the mimic response, users rarely wasted time rewording a request that was out of the scope of what the chatbot could handle.

6 Related Work

Verbal mimicry is used in conversation to build social rapport (Rogers, 1951; Rautalinko and Lisper, 2004). This observation has been leveraged even in the early development of conversational agents, for example in systems such as Eliza (Weizenbaum, 1966), which engaged users by picking up keywords and repeating open ended questions back to the patient, or PARRY (Colby et al., 1972), which follows a similar strategy to rephrase utterances to

[6]The live deployment was run at Eloquent Labs prior to their acquisition by Square Inc.

indicate anger or fear. Our work applies mimicry in the task-oriented setting and studies how and when generated mimic rephrasals are preferred over human responses.

More recently, sequence to sequence models have been used for open domain chatbots (Sordoni et al., 2015; Shang et al., 2015; Vinyals and Le, 2015; Wen et al., 2015; Serban et al., 2016). However, these models suffered from generic responses and turns that are semantically inconsistent and incoherent. To address these issues, Li et al. (2016a) introduced a maximum mutual information objective to encourage diversity. Consistency in dialog agents has been well studied in Kobsa and Wahlster (1989) *inter-alia*, and for neural methods by Li et al. (2016b). Rashkin et al. (2019) also recognized the need to acknowledge others' feelings in a conversation and introduced a dataset for benchmarking emphathetic dialog models.

Sequence-to-sequence models with copying was introduced in Gu et al. (2016). Such models have also been shown to be effective at semantic parsing (Jia and Liang, 2016), summarization (See et al., 2017; Cao et al., 2018), and task oriented dialog (Eric and Manning, 2017).

Our task can in many ways be considered a controlled generation task. Other work in this area includes generating text conditioned on a sentiment to express (Li et al., 2018), or controlled generation (Hu et al., 2017) by editing attributes (Shen et al., 2017; Logeswaran et al., 2018; Lample et al., 2019). These works can successfully change the tone and intent of an utterance, but tend to frequently rewrite enough of the content that the method is less effective for practical dialog applications. Ke et al. (2018) examined how to generate dialog responses with different sentence function (e.g., imperative, interrogative, etc.), which similarly allows for more distant rewriting than is optimal for our task. Finally, our task exhibits many of the challenges observed by Bilu et al. (2015) in the context of negating claims.

Other work in generation addresses tasks such as rephrasals for generating paraphrases (Prakash et al., 2016; Gupta et al., 2018), sentence simplification (Narayan et al., 2017), and query rewriting for question answering (Dong et al., 2017).

7 Conclusions

We proposed a new task and associated datasets for *mimicking and rephrasing* a speaker's prompt to communicate a given intent. We showed that both rule-based based and neural seq2seq models both approach human level performance. Additionally, we share observations from a real world deployment of the model to highlight how solving these tasks can potentially improve the end-user experience. We hope this will inspire future work in dialog agents, making these agents more fluent and personable.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*.

Yonatan Bilu, Daniel Hershcovich, and Noam Slonim. 2015. Automatic claim negation: Why, how and when. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Nathanael Chambers, Daniel Cer, Trond Grenager, David Hall, Chloe Kiddon, Bill MacCartney, Marie-Catherine De Marneffe, Daniel Ramage, Eric Yeh, and Christopher D Manning. 2007. Learning alignments and leveraging natural logic. In *ACL-PASCAL Workshop*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kenneth Mark Colby, Franklin Dennis Hilf, Sylvia Weber, and Helena C. Kraemer. 1972. Turing-like indistinguishability tests for the calidation of a computer simulation of paranoid processes. *Artificial Intelligence*, 3:199–221.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mihail Eric and Christopher Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *Proceedings of the Association for the Advancement on Artificial Intelligence (AAAI)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Liang Huang, Kai Zhao, and Mingbo Ma. 2017. When to finish? optimal beam search for neural text generation (modulo beam size). In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Pei Ke, Jian Guan, Minlie Huang, and Xiaoyan Zhu. 2018. Generating informative responses with controlled sentence function. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Alfred Kobsa and Wolfgang Wahlster. 1989. *User models in dialog systems*. Springer.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. *North American Association for Computational Linguistics (NAACL)*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*.

Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation with attribute controls. In *Advances in Neural Information Processing Systems*.

Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Shashi Narayan, Claire Gardent, Shay B Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*.

Aaditya Prakash, Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and

dataset. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Erik Rautalinko and Hans-Olof Lisper. 2004. Effects of training reflective listening in a corporate setting. *Journal of Business and Psychology*, 18(3):281–299.

Carl R. Rogers. 1951. *Client-centered therapy*. Riverside Press.

John R Searle. 1976. A classification of illocutionary acts. *Language in society*, 5(1):1–23.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Association for the Advancement on Artificial Intelligence (AAAI)*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of ACL-IJCNLP*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the International Conference on Machine Learning, Deep Learning Workshop*.

Joseph Weizenbaum. 1966. ELIZA —a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Automated Pyramid Summarization Evaluation

Yanjun Gao[1], **Chen Sun**[2], and **Rebecca J. Passonneau**[1]

[1,2]Department of Computer Science and Engineering
Pennsylvania State University
[1] {yug125,rjp49}@cse.psu.edu
[2]chensunx@gmail.com

Abstract

Pyramid evaluation was developed to assess the content of paragraph length summaries of source texts. A pyramid lists the distinct units of content found in several reference summaries, weights content units by how many reference summaries they occur in, and produces three scores based on the weighted content of new summaries. We present an automated method that is more efficient, more transparent, and more complete than previous automated pyramid methods. It is tested on a new dataset of student summaries, and historical NIST data from extractive summarizers.

1 Introduction

During the 70's and 80's, educational pychologists studied human summarization skills, and their development throughout secondary school and beyond. Three separate skills are acquired in the following order: selection of **important** information, **abstraction** through vocabulary generalization and sentence fusion, and **integration** with background knowledge (van Dijk and Kintsch, 1977; Brown and Day, 1983). A recent comparison of summaries from human experts versus extractive summarizers on forty-six topics from the TAC 2010 summarization challenge used automatic caseframe analysis, and found essentially these same properties in the human summaries, and not in the extractive ones (Cheung and Penn, 2013). Abstractive summarizers, however, are beginning to replicate the first two of these behaviors, as illustrated in many published examples based on encoder-decoder and pointer-generator neural architectures (Nallapati et al., 2016; See et al., 2017; Hsu et al., 2018; Guo et al., 2018). Summarization evaluation relies almost exclusively on ROUGE (Lin, 2004), an automated tool that cannot directly assess importance of summary content, or novel wording for the same information. We present an automated method to assess the importance of summary content, independent of wording, based on a widely used manual evaluation called pyramid (Nenkova et al., 2007).

Aligned PyrEval (W=5) and Manual (W=4) SCU	
RSUM1	**For example, an art gallery in London held an exhibit. with digital curr. as the preferred . . .**
RSUM2	However, there has been some positive **news as bus. such as a Scottish Hotel & a London Art Gallery are allowing cust. to pay with crypto currencies**
RSUM3	Cellan-Jones (2018) writes recent days **both a London art gallery and a Scottish hotel . . . to allow their cust. to pay with crypto-currencies.**
RSUM4	by suggesting that {a London art gallery & Scottish hotel chain plan to . . . support for different crypto-currencies.}$_{Paraph}$ . . . {*that the London based art gallery would use only crypto currencies*}$_{Paraph}$
RSUM5	*Businesses located in London and Scotland have made enquiries to allow payment from customers using cryptoc.*

Match to a student summary that used synomyms:
a *craftsmanship* exhibition alongside a Scottish *inn* have plans for their clients to pay in digital currencies

Figure 1: Alignment of a single PyrEval SCU of weight 5 to a manual SCU of weight 4 from a dataset of student summaries. The manual and automated SCUs express the same content, and their weights differ only by one. For each of five reference summaries (RSUM1-RSUM5), exact matches of words between the PyrEval and manual contributor are in bold, text in plain font (RSUM2, RSUM4) appears only in the manual version, and text in italics appears only in the PyrEval version. Paraphrases of the same content from RSUM4 were identified by human annotators (plain font) and PyrEval (italics). Also shown is a matching segment from a student summary, where the student used synonyms of some of the words in the reference summaries.

The pyramid method and ROUGE both use multiple summaries written by humans as references to assess new summaries. The manual pyramid method requires human annotators to identify Summary Content Units (SCUs) by grouping phrases from different reference summaries into

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 404–418
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

the same SCU if they express the same propositional content. Figure 1 illustrates an SCU from a manual pyramid applied to college student summaries of articles on cryptocurrency, with contributions from four of the five reference summaries (RSUM1-RSUM4). It is aligned to a nearly identical SCU constructed by PyrEval, with a contribution from the fifth reference (RSUM5). Previous work has shown that these kinds of discrepancies occur between human annotators, and have little effect on interannotator agreement or rankings of summarizers (Passonneau, 2010). The importance of an SCU increases with the number of reference summaries that express it, as indicated by its weight. If an evaluation summary expresses the same content as an SCU, its score is increased by the SCU weight (details below). ROUGE allows the user to select among numerous ways to measure ngram overlap of a new summary to the references, e.g., for different ngram sizes with or without skips, and with or without stemming. ROUGE does not, however, consider the relative importance of content, or account for synonyms of words that appear in the reference summaries.

We present PyrEval,[1] which outperforms previous work on automated pyramid in accuracy and efficiency. It produces human-readable pyramids, and prints matches between SCUs and evaluation summaries, which can support feedback for educational applications. PyrEval performs well on a new dataset of student summaries, where we applied the pyramid annotation. We also present results for TAC 2010 automated summaries, one of the more recent years where NIST applied pyramid evaluation. While ROUGE-2 more accurately identifies system differences than PyrEval, its performance is more sensitive to different topics.

2 Pyramid Content Analysis

The challenge in evaluation of summary content is that different equally good human summaries have only partial content overlap. van Halteren and Teufel (2003) annotated factoids (similar to FOL propositions, and to SCUs) for fifty summaries of a Dutch news article, and found a Zipfian distribution of factoid frequency: a small number of factoids represent 80% of the content in summaries, but a very long tail of rare content accounts for 20%. Pyramid annotation of ten summaries for a few DUC 2003 topics had a similar a Zipfian distribution (Nenkova and Passonneau, 2004).

Pyramid has had extensive reliability testing. A sensitivity analysis showed four reference summaries were sufficient for score reliability, and with probability of misranking errors below 0.01% (Nenkova and Passonneau, 2004; Nenkova et al., 2007). Interannotator agreement using Krippendorff's alpha on ten pyramids ranged from 0.61 to 0.89, and averaged 0.78 on matching new summaries to pyramids for 16 systems on 3 topics each (Passonneau, 2010). Comparison of two manual pyramid evaluations from distinct annotators showed that different pyramids for the same topic yield the same system rankings, even though SCUs from different pyramids typically do not align exactly (Passonneau, 2010).

The default size of a phrase that contributes to an SCU is a simple clause, but if it is clear from the context that a summary essentially expresses the same content expressed in other reference summaries, it is said to contribute to the same SCU, and the annotator must select at least a few contributing words. SCU weights reflect how many of $\mathcal{N}$ reference summaries express the SCU content. As such, SCUs are constrained to have no more than one contributor phrase from each reference summary. If a summary repeats the same information, the repetition will increment the count of total SCUs within one summary, but cannot be a distinct contributor. For example, the paraphrases from RSUM4 shown in Figure 1 add two to the total SCU size of the summary, but can only be used once to increment an SCU weight. Simple clauses in an evaluation summary that do not match pyramid SCUs add to the summary's SCU count, but have zero weight.

3 Related Work

Summarization is an important component of strategy instruction in reading and writing skills (Graham and Perin, 2007), but is used less than it could be due to the demands of manual grading and feedback. However, integration of NLP with rubric-based assessment has received increasing attention. Gerard et al. (2016) and Gerard and Linn (2016) applied automated assessment using rubrics to successfully identify students who need the most help, and facilitate and meaningful classroom interactions. Agejev and Šnajder (2017) used ROUGE and BLEU to identify col-

[1] Available at https://github.com/serenayj/PyrEval

lege students' L2 skills. Santamaría Lancho et al. (2018) used G-Rubric, an LSA-based tool, to help instructors grade short text answers to open-ended questions. Passonneau et al. (2018) found a high correlation of an automated pyramid with a manual rubric on a small set of summaries; see last paragraph of this section.

ROUGE is the most prevalent method to assess automated summarization. In 39 long papers on summarization in ACL conferences from 2013 through 2018 (mostly abstractive), 87% used ROUGE-1, ROUGE-2 or other variants such as LCS (longest common subsequence). A few used POURPRE (question answering) (Lin and Demner-Fushman, 2006), or METEOR (MT) (Denkowski and Lavie, 2014) to investigate scores based on weighted content or synonomy. POURPRE relies on string matching against reference units called answer facts, weighting matches by inverse document frequency. METEOR aligns words between reference and candidate, and can use relaxed word matching, such as WordNet synonymy. Despite its dominant use in previous work, Graham (2015) noted that the large range of ROUGE variants causes inconvenience and instability in evaluating performance. Graham's results from testing the 192 variants on DUC2004 data suggest that the ROUGE variants that correlate best with human evaluations are not often used.

PyrEval differs from other automated pyramid tools in its focus on accurately isolating and weighting the distinct SCUs in the reference summaries. Three previous semi-automated pyramid tools used dynamic programming to score summaries, given a manual pyramid (Harnly et al., 2005; Passonneau et al., 2013, 2018). The first of these used unigram overlap to compare summaries to a pyramid. Absolute scores were much lower than ground truth, but average system rankings across multiple tasks were accurate. A subsequent extension that used cosine similarity of latent vector representations of ngrams and SCUs, based on (Guo and Diab, 2012), had much better performance (Passonneau et al., 2013). This was extended further through use of a weighted set cover algorithm for scoring (Passonneau et al., 2018). PEAK was the first fully automated approach to construct a pyramid and score summaries (Yang et al., 2016). It uses OpenIE to extract subject-predicate-object triples from references, then constructs a hypergraph with triples as hyperedges. Semantic similarity between nodes from distinct hyperedges is measured using ADW's random walks over WordNet (Pilehvar et al., 2013), to assign weights to triples. On a small set of summaries used here in Table 1, PEAK raw scores had a high correlation with a manual summary rubric. PEAK was also tested on a single DUC 2006 topic, where the input text was first manually altered. Because PEAK is slow, Peyrard and Eckle-Kohler (2017) reimplemented it's use of the Hungarian algorithm to optimize their summarizer. Because PEAK produces many noisy copies of the same SCU, its output cannot be used to justify scores based on the unique matches or misses of a student summary to SCUs. Its score normalizations are inaccurate, and the un-normalized scores are impractical for general-purpose evaluation.

4 PyrEval System

To construct a pyramid, humans identify contributor segments[2] and group them into SCUs. Evaluating a summary is a simpler process of matching phrases to existing SCUs. PyrEval performs analogous steps, as shown in Figure 2. It first decomposes sentences of reference summaries (RSUM) into segments (DECOMP PARSE) and converts them into semantic vectors (LATENT SEM). It then applies EDUA to group the segment vectors into SCUs. EDUA (see below) is a novel restricted set partition algorithm that maximizes the semantic similarity within SCUs, subject to SCU constraints. Evaluation summaries (ESUM) are preprocessed in a similar fashion to convert them to segments represented as vectors. As in (Passonneau et al., 2018), PyrEval applies WMIN (Sakai et al., 2003) to find the optimal set of matches with pyramid SCUs. The remainder of this section describes each step.

4.1 Sentence Decomposition

The decomposition parser takes as input a phrase structure parse and dependency parse for each sentence, using Stanford CoreNLP (Manning et al., 2014). Every tensed verb phrase (VP) from the phrase structure parse initializes a new segment. The head verbs of tensed VPs are aligned to the dependency parse, and their dependent subjects are then attached to the segments. Words other than

[2]These can be discontinuous substrings, and can reuse words from other contributors, e.g., subjects of VP conjuncts.

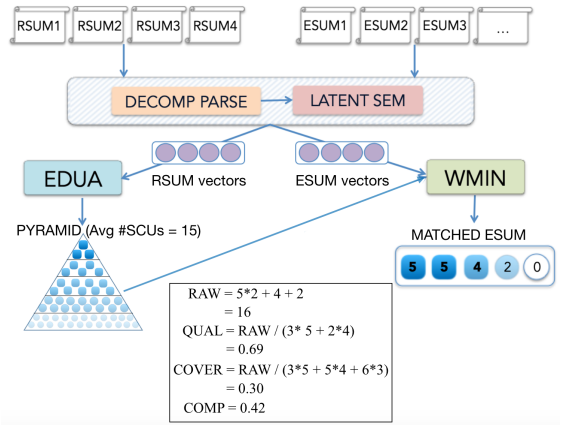


Figure 2: PyrEval preprocessors segment sentences from reference (RSUM) and evaluation (ESUM) summaries into clause-like units, then convert them to latent vectors. EDUA constructs a pyramid from RSUM vectors (lower left): the horizontal bands of the pyramid represent SCUs of decreasing weight (shaded squares). WMIN matches SCUs to ESUM segments to produce a raw score, and three normalized scores.

Data (size)	ELMo	USE	GloVe	WTMF
WIM (20)	0.3873	-0.0290	0.7149	**0.8525**
TAC 09 (54)	0.0515	0.2672	0.1713	**0.4961**
STS-14 (3750)	0.1636	0.5757	0.6129	**0.7257**

Table 1: Comparison of phrase embedding methods on correlation to manual pyramd (WIM, TAC09) or correlation to human similarity judgements (STS-14).

those in the VP and subject are reinserted in their original order. Every sentence has at least one default segmentation corresponding to the full sentence, possibly with one or more alternative segmentations of at least two segments each. It performs well for most cases apart from sentences with coordinate structures, which are notoriously difficult for conventional parsers. Figure 3 illustrates a sentence segmentation, with three alternatives.[3]

4.2 Semantic Vectors for Segments

The second PyrEval preprocessing step converts segments to semantic vectors. We chose to avoid semantic representation that requires training, to make PyrEval a lightweight, standalone tool. Although recent contextualized representations perform very well on a variety of NLP tasks, they are typically intended as the basis for a transfer learning approach, or to initialize further task-specific

[3]Segmentation 1.6.2 is the one EDUA-G selects for the pyramid.

Christopher Shake, the director of the London art gallery, suggests that this is not the case and that many different companies of different natures not just technology related are getting involved with cryptocurrencies.

1.6.1.0 that this is not the case
1.6.1.1 Christopher Shake, the director of the London art gallery, suggests and.
1.6.1.2 that many different companies of different natures not just technology related are getting involved with cryptocurrencies
1.6.2.0 Christopher Shake, the director of the London art gallery, suggests that this is not the case and.
1.6.2.1 that many different companies of different natures not just technology related are getting involved with cryptocurrencies
1.6.3.0 Christopher Shake, the director of the London art gallery, suggests and that many different companies of different natures not just technology related are getting involved with cryptocurrencies.
1.6.3.1 that this is not the case

Figure 3: Segmentation output for a sentence from a reference summary for the "CryptoCurrencies" topic of our student summaries.

neural training (e.g., (Pagliardini et al., 2018; Peters et al., 2018; Devlin et al., 2018; Vaswani et al., 2017)). The most practical way to rely on completely pre-trained representations is to use word embeddings along with a method to combine them into phrase embeddings. Here we report on a comparison of ELMo (Peters et al., 2018) and the Universal Sentence Encoder for English (USE) (Cer et al., 2018) with two conventional word embedding methods, GloVe (Pennington et al., 2014) and WTMF (Guo and Diab, 2012).[4]

ELMo is character-based rather than word-based, relies on a many-layered bidirectional LSTM, and incorporates word sequence (language model) information. It was trained on billions of tokens of Wikipedia and news text. To create meaning vectors for strings of words, we use pre-trained ELMo vectors, taking the weighted sum of 3 output layers as the word embeddings, then applying mean pooling.[5] USE is intended for transfer learning tasks, based on Transformer (Vaswani et al., 2017) or the (Iyyer et al., 2015) deep averaging network (DAN). We create meaning vectors for word strings with the USE-DAN pretrained encoder.[6] We use the GloVe download for 100D vec-

[4]We do not show results for Word2Vec (Mikolov et al., 2013), where performance was similar to GloVe.

[5]We use ELMo module from https://github.com/allenai/allennlp/.

[6]https://tfhub.dev/google/universal-sentence-encoder/2.

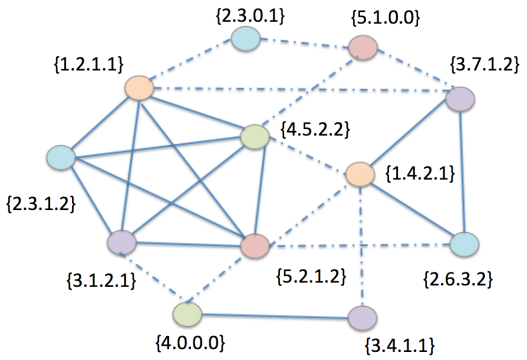


Figure 4: Part of an EDUA solution graph. Each vertex is a segment vector from a reference summary, indexed by **Summary.ID** (s_i)**, Sentence.ID** (s_{ij})**, Segmentation.ID** (s_{ijk})**, Segment.ID** (s_{ijkm}). All segments of all reference summaries have a corresponding node. All edges connect segments from different summaries with similarity $\geq t_{edge}$. This schematic representation of a partial solution contains three fully connected subgraphs with attraction edges (solid lines), each representing an SCU, whose weight is the number of vertices (segments).

1. Given a set of n reference summaries R, a preprocessing function (described in subsections 4.1-4.2) SEG returns segments as vectors: $\forall R_i \in R$, SEGS$(R_i) = \{seg_{ijk1}, seg_{ijk2}, \ldots, seg_{ijkm}\}$ where seg_{ijkm} is the mth segment of the kth segmentation of the jth sentence in the ith summary.
2. A graph G is constructed from SEGS(R_i), where an edge connects segments $seg_{ijkm}, seg_{i'j'k'm'}$ if $(i \neq i', seg_{ijkm} \neq seg_{i'j'k'm}$, cosine$(seg_{ijkm}, seg_{i'j'k'm}) \geq t_{edge})$. Every fully connected subgraph (clique) is a candidate scu whose size is the number of nodes, which has a maximum of n.
3. The attraction score of an $scu^z, \mathcal{AS}(scu^z) = \frac{1}{\binom{|scu^z|}{2}} \sum_{seg_{ijkm}, seg_{i'j'k'm} \in scu^z, seg_{ijkm} \neq seg_{i'j'k'm'}}$ cosine$(seg_{ijkm}, seg_{i'j'k'm'})$ if $z > 1$, else $= 1$.
4. A candidate pyramid P is a set of equivalence classes SCU^x that is a covering of all sentences in R (meaning only one segmentation per sentence belongs to any P), $\forall\ x \in [1,n] : (\exists\ SCU^x \in P) \rightarrow (x \in [1,n], \forall scu^z \in SCU^x, x = z)$. An SCU^x has an attraction class score $\mathcal{AC}(SCU^x) = \frac{1}{|SCU^x|} \sum_{scu^z \in SCU^x} \mathcal{AS}(SCU^z)$.
5. Finally, a pyramid P has an attraction score $\mathcal{AP}(P) = \sum_{SCU^x \in P} \mathcal{AS}(SCU^x)$.
6. The optimal pyramid(R) = $\mathcal{P}$ that maximizes $\mathcal{AP}$.

Figure 5: Formal specification of EDUA's input graph G consisting of all segments from all segmentations of reference summary sentences (item 2), the objective (item 6), and three scores for defining the objective function that are assigned to candidate SCUs (item 3), sets of SCUs of the same weight (item 4), and a candidate pyramid (item 5).

tors trained on the 840B Common Crawl.[7] To combine the GloVe word vectors into a phrase vector, we use the weighted averaging method from (Arora et al., 2016). WTMF is a matrix factorization method. We use WTMF matrices trained on the Guo and Diab (2012) corpus (393K sentences, 81K vocabulary size) that consists of WordNet, Wiktionary, and the Brown corpus.

We compare the four embedding methods on three datasets. Because our goal is to select a method that performs well on pyramid annotation, the first two datasets are human and machine summaries with manual pyramid annotations, with correlation of the manual pyramid and PyrEval scores as the metric. WIM (for *What is Matter*) is a dataset of student summaries with pyramid annotation from (Passonneau et al., 2018) with 20 student summaries on one topic. Note that PyrEval achieved a correlation of 0.85 on this data, compared with 0.82 for PEAK (Passonneau et al., 2018). We also use a subset of data from the NIST TAC 2009 summarizer challenge. We use summaries from all 54 peer systems on 14 of the 44 topics. We also use the STS-14 benchmark dataset of semantic similarity judgements (3750 sentence pairs), as in (Guo and Diab, 2012).

Table 1 shows WTMF to perform best on the three tasks by a large margin. We speculate this results from two factors. The lower dimensionality of WTMF vectors compared to ELMo or USE-DAN leads to higher maximum cosine values, thus better contrast between similar and dissimilar pairs. WTMF differs from similar matrix reduction methods in assigning a small weight to non-context words, which improves robustness for short phrases (fewer context words) Guo and Diab (2012). The authors also claimed that a training corpus largely consisting of definitional statements leads to co-occurrence data that is less noisy than sentences found in the wild.

4.3 EDUA

EDUA (Emergent Discovery of Units of Attraction) is a restricted set partition algorithm. It constructs an optimal pyramid to achieve the highest attraction (semantic similarity of segments) in all SCUs. Figure 4 schematically represents the input graph to EDUA (see also item 1 in Fig-

[7] https://nlp.stanford.edu/projects/glove/.

ure 5), whose nodes consist of the segment vectors described in the preceding section, and whose edges connect segments from different summaries whose cosine similarity $\geq t_{edge}$.[8] A candidate SCU is a fully connected subgraph (clique; item 2 in Figure 5). Every candidate SCU has an attraction score $\mathcal{AS}$ equal to the average of the edge scores (item 3 in Figure 5). A candidate pyramid is a set of SCUs that constitute a covering of all the sentences in the input reference summaries, with all segments for a given sentence coming from only one of its segmentations. The SCU weights for a pyramid, which are in $[1, n]$ for n reference summaries, form a partition over its segments, and each equivalence class (all SCUs of the same weight) has a score $\mathcal{AC}$ that is the average of its SCU scores (item 4 in Figure 5). The score for a candidate pyramid $\mathcal{AP}$ is the sum of its $\mathcal{AC}$ scores (item 5 in Figure 5). We use the sum rather than the average for $\mathcal{AP}$ to favor the equivalence classes for higher weight SCUs. The optimal pyramid maximizes $\mathcal{AP}$ (item 6 in Figure 5).

EDUA has two implementations. EDUA-C implements a complete solution based on depth first search (DFS) of candidate SCUs that guarantees the global optimum (max $\mathcal{AP}$). EDUA-G is a greedy approximation.[9]

4.4 EDUA-C

EDUA-C constructs an adjacency list that for each clique (candidate SCU) in the input graph, identifies all the other SCUs that satisfy two constraints: 1) for any given sentence, all SCUs reference the same segmentation; 2) all segments in all SCUs are distinct. DFS search proceeds through the adjacency list, ordering the SCUs by weight, until a path is found through all SCUs that meets the constraints. The solution has the highest $\mathcal{AP}$, or in the case of ties, the path found first.

Figure 6 illustrates a toy EDUA-C DFS tree. Each node depicts a candidate SCU clique, labelled by the number of nodes in the clique (SCU weight). No child node has a higher weight than its parent nodes. A child node is added to a search path (solid nodes) if it violates no constraints. Each of the six paths in the figure would receive an $\mathcal{AP}$ score. After DFS finds all legal paths, the one with highest $\mathcal{AP}$ is selected as the solution.

[8] The value of t_{edge} is automatically set to the 83rd percentile of all pairwise cosine similarities in the input data.

[9] See Appendix A for EDUA-G.

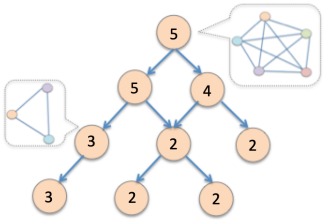


Figure 6: A directed Depth First Search tree for EDUA-C. Nodes are cliques representing candidate SCUs, as illustrated in Figure 4, labeled by their weights. Each DFS path is a partition over one way to segment all the input summaries and group all segments into SCUs. The solution is the path with the highest $\mathcal{AP}$.

4.5 Comparison of EDUA variants

Table 2 compares the distribution of SCUs by weight of the two EDUA variants with manual pyramids on the student summary dataset discussed in the next section. EDUA-C produces a more skewed distribution than EDUA-G. Both variants suffer from the coarse-grained segmentation output from the decomposition parser, but EDUA-G compensates by enforcing the Zipfian distribution observed in most pyramids (see appendix A for details).

To evaluate speed, we tested both variants on datasets with different numbers and lengths of reference summaries. TAC 2010 reference summaries (4 per topic) have on average 46 segments each, and 321 candidate SCUs. Pyramid construction for TAC 2010 takes less than 10 seconds with either variant on an Ubuntu machine with 4 Intel i5-6600 CPUs. EDUA-G's greater efficiency is more apparent for larger input. DUC 2005 has seven reference summaries per topic, and longer summaries than in TAC 2010; on five, EDUA-C takes 211 seconds, while EDUA-G is still only about ten seconds; on six, EDUA-C takes 20 minutes, compared to 5 minutes for EDUA-G.

Topic	Variant	All	w=5	w=4	w=3	w $\leq$ 2
CC	Manual	34	0	3	5	26
	G	31	1	2	4	24
	C	39	1	1	1	36
AV	Manual	41	0	6	2	33
	G	29	0	1	4	24
	C	35	1	1	1	32

Table 2: Comparison of distributions of SCUs by weight from pyramids produced manually, by two EDUA variants (G and C), for the two topics CC and AV.

4.6 WMIN Scoring

For automatic matching of phrases in evaluation summaries to SCUs in a manual pyramid, Passonneau et al. (2018) found good performance with WMIN (Sakai et al., 2003), a greedy maximum weighted independent set algorithm. Because EDUA pyramids are analogous to manual pyramids, PyrEval also uses WMIN. The input to WMIN is a graph where each node is a tuple of a segmentation of an ESUM sentence with the sets of SCUs that give the highest average cosine similarity for that sentence. The node weight is the sum of SCU weights. Graph edges enforce constraints that only one segmentation for a sentence can be selected, and each pyramid SCU can be matched to an ESUM sentence at most once. WMIN selects the nodes that result in the maximum sum of SCU weights for the ESUM.

Score computation is a function of the matched SCUs, as illustrated by the ESUM in the lower right of Figure 2. This ESUM has five SCUS: two of weight 5, one of weight 4, one of weight 2, and one that does not match the pyramid (zero weight). The sum of SCU weights is 16. The original pyramid score, a precision analog, normalizes the raw sum by the maximum sum for the same SCU count given by the pyramid – $(3 \times 5) + (2 \times 4)$ – indicating the degree to which the summary SCUs are as high weighted as possible. Following (Passonneau et al., 2018), we use the term quality score. The average number of SCUs in the reference summaries is 15, whose maximum weight from this pyramid is 53. Normalizing the raw sum by 53 gives a coverage score of 0.30 (a recall analog). The harmonic mean of these scores gives an F score analog referred to as a comprehensive score.

5 Student Summaries

As part of a collaboration with a researcher in educational technology, we collected a new data set of student summaries that were assigned in fall 2018 to computer science freshman in a university in the United Kingdom (Gao et al., 2019). Our immediate goal is to see how PyrEval could support instructors who assign summaries by providing an automated assessment that could be later corrected, but which provides scores and score justifications. PyrEval scores correlate well with manual pyramid scores on content, and the log output it produces provides a clear trace of score computation (see below).

Topic	Variant	Raw/Cov.	Qual.	Comp.	R2
AV	EDUAG	**0.66**	0.48	0.56	0.61
	EDUAC	0.55	0.50	0.53	
CC	EDUAG	**0.72**	0.63	0.69	0.66
	EDUAC	0.55	0.48	0.51	

Table 3: Pearson correlation of manual pyramid and PyrEval on four scores (raw/coverage, quality and comprehensive) compared with ROUGE-2 on coverage.

The class was an academic skills class that included instruction in academic reading and writing. For one assignment, they were instructed to select one of two current technology topics (three readings per topic), then to **summarize** it in 150 to 250 to words. The two topics are shown below, with the number of student summaries per reading, and average number of words.

1. Autonomous Vehicles (AV): 42 summaries, average words = 237.76
2. Cryptocurrency (CC): 37 summaries, average words = 245.84

To write reference summaries for both topics, the instructor recruited advanced students who had done well in her academic skills class in previous years. Three trained annotators applied manual pyramid annotation to the student summaries. As noted in section 2, pyramid annotation is highly reliable. Annotations of the student summaries were performed in two passes by different annotators.

Table 3 reports the correlation between the manual pyramid scores and the PyrEval scores on the two sets of student summaries. For both AV and CC, EDUA-G performs better than EDUA-C and ROUGE-2, the best ROUGE variant on TAC10 (see below), and ROUGE-2 performs better than EDUA-C. We attribute the lower correlations on the quality score, and the lower performance on this dataset compared to WIM (see Table 1), to the greater challenges of the new dataset. WIM students read a single, middle school text, and average summmary length was 109.02 words. For the new dataset, students read three advanced texts, and produced summaries that were over twice the length (see above). Error analysis shows complex sentence structure for the AV and CC data, with many constructions such as conjunctions and lists, that the decomposition parser cannot handle. As noted above, EDUA-G compensates due to a Zipfian constraint on the pyramid shape.

Figure 1 compares a PyrEval SCU with a manual one for the cryptocurrency topic, and

Single PyrEval SCU (W=3) about the relation of "car accidents" to "insurance cost"	
RSUM1	Also, as most collisions are due to human error, costs of insurance for self driving cars could fall by up to <NUM>.
RSUM2	The cars themselves would also reduce insurance premiums; <NUM> percent of road accidents are caused by human error
RSUM3	Shankleman does well to balance out the positives such as lower insurance , reduced traffic , savings on mechanical costs and lower chance of road accidents .

	Single manual SCU (W=4) on "high car accidents"	Single manual SCU (W=4) on "lower insurance"
RSUM1	Also, as most collisions are due to human error	costs of insurance for self-driving cars could fall by up to 50%
RSUM2	90 percent of road accidents are caused by human error	The cars themselves would also reduce insurance premiums
RSUM3	. . . lower chance of road accidents . . .	. . . lower insurance . . .
RSUM4	. . . he claims that over 90 percent of road traffic accidents occur as a result of human error	*this* would result in lower insurance premium for owners of autonomous vehicles by up to 50 percent.

Student IDs	Segments correctly matching this PyrEval SCU to students' summaries (from PyrEval log output)
A	The **insurance industry is also going to experience great changes** as the director insurer of AXA SA explains that more than <NUM> percent of **road accidents are caused by human error**.
B	as <NUM> of the **accidents are caused by human errors**, also reducing the number of human drivers will contribute to **cheap insurance premiums** and efficient transport
C	Shankleman explains how problems with modern day transport such as high **crash statistics and extortionate insurance costs will be eradicated** with such computing capabilities.

Figure 7: Alignment of an PyrEval SCU of weight 3 to segments from student summaries on autonomous vehicle.

also illustrates issues that might explain the relatively poorer performance of ROUGE. We show a phrase that both the manual annotator and PyrEval matched to the SCU from one of the student summaries, where the student used near synonyms for terms in the articles and reference summaries: *craftmanship* exhibition for *art gallery*, and *inn* for *hotel*. ROUGE cannot match synonyms, and does not distinguish differences in content importance.

Figure 7 shows an excerpt from PyrEval's log output on autonomous vehicle to illustrate the alignment of an SCU to three student summaries and comparison to two manual SCUs.[10] The PyrEval SCU captures a causal relation between "car accidents due to human error" and "lower insurance costs." The two manual SCUs, however, show that the human annotators split this content into two SCUs, because the content is expressed in distinct clauses in RSUM1 and RSUM2. The same content is supported by the implicit contexts for the shorter RSUM3 contributing phrases. The RSUM4 contributor in the manual SCU about "lower insurance" illustrates another issue that PyrEval preprocessing cannot handle: resolution of the deictic pronoun subject in "this would result . . . ".

[10] Preprocessing replaces numeric character strings with tags.

6 TAC 2010 Summaries

NIST summarization challenges dealt exclusively with news, which is also the most prevalent genre for automated summarizers in our survey of 2013-2018 ACL publications (23/39 summarizers; see above). To evaluate ROUGE, NIST used two human gold standards in yearly challenges from 2005 through 2011, one of which was manual pyramid. Annotation was performed by volunteers among the challenge participants, using guidelines developed for DUC 2006.[11] In this section, we apply a method NIST helped develop to evaluate ROUGE against manual pyramid in an evaluation of PyrEval against manual pyramid. We selected TAC 2010 because summarizer performance was less good in the earlier years.

TAC 2010 had two 100-word summarization tasks on 10 documents for 46 topics. Task A summarization was guided by a query. Task B was an update to A, based on additional input. On inspection of the 92 pyramids (46 each for Tasks A and B), we found that roughly 27% had poor quality pyramids that did not follow the guidelines mentioned above. We assembled a team of five people familiar with manual pyramid to manually redo the twelve pyramids that were independently identified as the lowest quality.[12]

Tests of the correlation of human scores as-

[11] http://www1.cs.columbia.edu/~becky/DUC2006/2006-pyramid-guidelines.html; we followed these guidelines for annotating the student

System	Task A			Task B		
	Mean Acc. (sdev)	95% CI	Acc. on 46 (delta)	Mean Acc. (sdev)	95% CI	Acc. on 46
R1	0.70 (0.04)	(0.69, 0.71)	*0.73 (0.03)*	0.61 (0.04)	(0.60, 0.62)	*0.69 (0.08)*
R2	**0.72 (0.03)**	(0.72, 0.73)	*0.80 (0.08)*	**0.70 (0.05)**	(0.69, 0.71)	*0.78 (0.08)*
EDUA-C	0.57 (0.04)	(0.57, 0.58)	0.65 (0.08)	0.56 (0.05)	(0.55, 0.57)	0.62 (0.06)
EDUA-G	0.57 (0.03)	(0.56, 0.57)	0.60 (0.04)	0.60 (0.03)	(0.59, 0.60)	0.63 (0.04)

Table 4: Mean accuracy, standard deviation and 95% confidence intervals on TAC 2010 Wilcoxon results for ROUGE-1, ROUGE-2 and PyrEval, using 100 bootstrapped samples of 41 of the 46 topics.

signed to automated summaries with ROUGE (and other automated metrics) were found to be unreliable, because of high score variance resulting as much from properties of the input texts as from differences in summarization systems (Nenkova, 2005; Nenkova and Louis, 2008). Analyses of over a decade of NIST data from automated summarizers that evaluate ROUGE against manual pyramid and another manual score led to a solution to this problem (Rankel et al., 2013; Owczarzak et al., 2012a,b; Rankel et al., 2011). The solution is to use Wilcoxon signed rank tests, so that pairs of systems are compared on matched input in a way that tests for statistical significance. The outcome is either that one of the systems is significantly better than the other, or that the difference between them is not statistically significant. To determine if the automated metric accurately reflects the gold standard scores, the same Wilcoxon tests are performed using the manually assigned scores on all pairs of systems, matching each pair on the same inputs. A given automated metric is then compared to the human gold standard to determine how accurately the automated metric leads to the same set of significant differences between all pairs of systems.

Table 4 presents bootstrapped accuracy results for ROUGE and PyrEval using 41 topics per bootstrap sample, along with absolute accuracy on all 46 topics. Each selection of 41 topics gives a gold standard set of system differences against which to compare a given metric. ROUGE 2 has the highest average accuracy on both Task A and B. ROUGE 1 performs nearly as well on Task A. PyrEval performs less well on average accuracy for all tasks, but similarly to ROUGE 1 in Task B. ROUGE-2 has greater sensitivity to topics, as shown by the higher deltas between the bootstrapped accuracy on 41 topics versus the accuracy on all 46. The differences in Table 4 between the bootstrapped averages across 41 topics, and the accuracy scores on all 46 topics, confirms the sensitivity of evaluation results to topics noted in (Nenkova, 2005; Nenkova and Louis, 2008).

7 Conclusion

PyrEval outperforms previous automated pyramid methods in accuracy, efficiency, score normalization, and interpretability. It correlates with manual pyramid better than ROUGE on a new dataset of student summaries, and produces output that helps justify the scores (similar to the examples for Figures 1 and 7). While it does not perform as well as ROUGE on extractive summarization, we speculate it would outperform ROUGE on abstractive summarizers. It relies on EDUA, a novel restricted set partition algorithm, that expects semantic vectors of sentence segments as input. The current rule-based method that identifies sentence substrings (the decomposition parser) is limited by the output of the constituency and dependency parsers it relies on. We are currently working on a neural architecture that simultaneously identifies simple clauses and produces semantic representations that could provide better input for both EDUA and WMIN, and thus improve PyrEval.

8 Acknowledgements

This work was supported in part by a Fellowship from Teaching and Learning with Technology, Penn State University, and by NSF award IIS-1847842. We thank two Penn State undergraduate research assistants for their contributions to the code base: Andrew Warner, and Purushartha Singh. Brent Hoffert, who recently graduated from Penn State, developed the wrapper that simplifies the use of PyrEval. Several additional Penn State undergrads helped correct the TAC 10 pyramids: Brent Hoffert, Alex Driban, Sahil Mishra, Xuannan Su, and Kun Wang. Finally, we thank the reviewers for their helpful suggestions.

summaries.

[12]We plan to ask NIST if we can make this data available through them.

References

Tamara Sladoljev Agejev and Jan Šnajder. 2017. Using analytic scoring rubrics in the automatic assessment of college-level summary writing tasks in l2. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 181–186.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations (ICLR 2017)*.

Ann L. Brown and Jeanne D. Day. 1983. Macrorules for summarizing texts: The development of expertise. *Journal of Verbal Learning and Verbal Behavior*, 22:1–14.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Jackie Chi Kit Cheung and Gerald Penn. 2013. Towards robust abstractive multi-document summarization: A caseframe analysis of centrality and domain. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1233–1242. Association for Computational Linguistics.

Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. 2009. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation (ACL)*, pages 376–380, Baltimore, MD.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Teun A. van Dijk and Walter Kintsch. 1977. Cognitive psychology and discourse: Recalling and summarizing stories. In W. U. Dressier, editor, *Trends in textlinguistics*, pages 61–80. De Gruyter, New York.

Yanjun Gao, Alex Driban, Brennan Xavier McManus, Elena Musi, Patricia Davies, Smaranda Muresan, and Rebecca J Passonneau. 2019. Rubric reliability and annotation of content and argument in source-based argument essays. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 507–518.

Libby Gerard, Marcia C Linn, and Jacquie Madhok. 2016. Examining the impacts of annotation and automated guidance on essay revision and science learning. In C. K. Looi, J. L. Polman, U. Cress, and P. Reimann, editors, *Transforming Learning, Empowering Learners: The International Conference of the Learning Sciences (ICLS) 2016*. Singapore: International Society of the Learning Sciences.

Libby F Gerard and Marcia C Linn. 2016. Using automated scores of student essays to support teacher guidance in classroom inquiry. *Journal of Science Teacher Education*, 27(1):111–129.

Steve Graham and Dolores Perin. 2007. A meta-analysis of writing instruction for adolescent students. *Journal of Educational Psychology*, 99(3):445–476.

Yvette Graham. 2015. Re-evaluating automatic summarization with BLEU and 192 shades of ROUGE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 128–137, Lisbon, Portugal. Association for Computational Linguistics.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 687–697. Association for Computational Linguistics.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th ACL*, pages 864–872.

Hans van Halteren and Simone Teufel. 2003. Examining the consensus between human summaries: Initial experiments with factoid analysis. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Summarization*, pages 57–64. Association for Computational Linguistics.

Aaron Harnly, Ani Nenkova, Rebecca J. Passonneau, and Owen Rambow. 2005. Automation of summary evaluation by the pyramid method. In *Proceedings of the Conference of Recent Advances in Natural Language Processing (RANLP)*, pages 226–232.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 132–141. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the*

7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8, pages 74–81. Barcelona, Spain.

Jimmy Lin and Dina Demner-Fushman. 2006. Methods for automatically evaluating answers to complex questions. *Information Retrieval*, 9:565–587.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *20th SIGNLL Conference on Computational Natural Language Learning (CoNLL*, pages 280–290. ACL.

Ani Nenkova. 2005. Automatic text summarization of newswire: Lessons learned from the document understanding conference. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3*, AAAI'05, pages 1436–1441. AAAI Press.

Ani Nenkova and Annie Louis. 2008. Can you summarize this? identifying correlates of input difficulty for multi-document summarization. In *Proceedings of ACL-08: HLT*, pages 825–833, Columbus, Ohio. Association for Computational Linguistics.

Ani Nenkova and Rebecca J. Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *HLT-NAACL 2004*.

Ani Nenkova, Rebecca J. Passonneau, and Kathleen McKeown. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(2):4.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012a. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Karolina Owczarzak, Hoa Trang Dang, Peter A. Rankel, and John M. Conroy. 2012b. Assessing the effect of inconsistent assessors on summarization evaluation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 359–362. Association for Computational Linguistics.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.

Rebecca J. Passonneau. 2010. Formal and functional assessment of the pyramid method for summary content evaluation. *Nat. Lang. Eng.*, 16(2):107–131.

Rebecca J. Passonneau, Emily Chen, Weiwei Guo, and Dolores Perin. 2013. Automated pyramid scoring of summaries using distributional semantics. In *ACL*, pages 143–147.

Rebecca J. Passonneau, Ananya Poddar, Gaurav Gite, Alisa Krivokapic, Qian Yang, and Dolores Perin. 2018. Wise crowd content assessment and educational rubrics. *International Journal of Artificial Intelligence in Education*, 28(1):29–55.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Maxime Peyrard and Judith Eckle-Kohler. 2017. Supervised learning of automatic pyramid for optimization-based multi-document summarization. In *ACL 2017*, pages 1084–1094.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *ACL*, pages 1341–1351.

Peter Rankel, John M. Conroy, Eric V. Slud, and Dianne P. O'Leary. 2011. Ranking human and machine summarization systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 467–473, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter A. Rankel, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2013. A decade of automatic content evaluation of news summaries: Reassessing the state of the art. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 131–136, Sofia, Bulgaria. Association for Computational Linguistics.

Shuichi Sakai, Mitsunori Togasaki, and Koichi Yamazaki. 2003. A note on greedy algorithms for the maximum weighted independent set problem. *Discrete Applied Mathematics*, 126(2):313–322.

Miguel Santamaría Lancho, Mauro Hernández, Ángeles Sánchez-Elvira Paniagua, José María Luzón Encabo, and Guillermo de Jorge-Botana. 2018. Using semantic technologies for formative assessment and scoring in large courses and MOOCs. *Journal of Interactive Media in Education*, 2018(1).

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Qian Yang, Rebecca J. Passonneau, and Gerard de Melo. 2016. PEAK: Pyramid evaluation via automated knowledge extraction. In *AAAI*, pages 2673–2680.

A EDUA-G

EDUA-G (*Greedy*) is a greedy approximation to EDUA-C with a backtracking algorithm adjusting the allocation of candidate SCUs, and enforcing the constraints. In this appendix, we use the same notation as in section 4.3. Instead of finding the solution globally with maximum $\mathcal{AP}$ across all possible pyramids, EDUA-G works on achieving the maximum $\mathcal{AS}$ for each set of SCUs of a given size locally, starting with the set C_n with highest weight (number of nodes per subgraph c), then the rest in descending order. In addition to the constraints 1 and 2 in EDUA-C (mentioned in section 4.4), EDUA-G has a capacity constraint for each set C_r during search, limiting the number of SCUs committed to the class. This constraint is determined by the length of all the reference summaries and exploits an empirical observation of pyramids: that SCUs have a Zipfian distribution of frequency across reference summaries: a few have the highest weight, and for each lower weight there are more in number, with a very long tail of SCUs of weight 1.

To enforce the capacity constraint during search, we define the maximum number of SCUs y_n of each equivalence class C_n as:

$$y_n = \alpha\left(\frac{1}{n}\right)^{\beta} \tag{1}$$

where n is the index of the equivalence class, α is a constant related to the total number of segments from all reference summaries, and β is a scaling parameter (Clauset et al., 2009). Thus in addition to t_{edge}, EDUA-G has the hyperparameters α and β. The capacities of the equivalence classes are monotone increasing as n decreases:

$$|C_n| \leq |C_{n-1}| \tag{2}$$

Summing over $|C_n|$ gives the size of the pyramid:

$$\sum_{i}^{N} |C_n| \leq \sum_{i}^{N} y_i \tag{3}$$

Algorithm 1 presents EDUA-G.

Initialization Similar to EDUA-C, a segment pool $SP = SEGS(R_1) \cup \ldots \cup SEGS(R_n)$ is first constructed from all the reference summaries to store segments and two status flags. The pool is accessible globally. For every segment seg_{ijkm}, two status flags are set:

Algorithm 1: EDUA-G

Data: Number of reference summaries n; a list CU of candidate SCUs ordered by weight r where $1 \leq r \leq n$, then by attraction score $\mathcal{AS}(CU^r)$; capacity of each equivalence class $y_1 \ldots y_n$ by formula 1; a segment pool SP, residuals L_1

Result: Pyramid P with equivalence classes $C_1 \ldots C_n$

1 Initialize $r = n$,
2 $C_r = \emptyset$, $P = \emptyset$, D_r as empty stack,
3 **while** $(r > 1) \wedge (|C_r| \leq y_r)$ **do**
4 push all candidate CU^r selected from CU into D_r sorted by attraction score in ascending order ;
5 **while** D_r *is not empty* **do**
6 pop e from D_r with maximum $\mathcal{AS}$;
7 **if** *notConflict*(C_r, e), *and* $\forall\ seg_{ijkm} \in e$ $seg_{ijkm}.commit == True$ *or* $seg_{ijkm}.commit == NotValid$, *and* $seg_{ijkm}.used == False$ **then**
8 Commit(C_r, e, SP) ;
9 **end**
10 **end**
11 **if** *P fails to meet any of the constraints* **then**
12 BackTrack$(C_r, y_r, C_{r+1}, P, SP)$
13 **else**
14 $P \leftarrow P \cup C_r$;
15 **end**
16 $r \leftarrow r - 1$;
17 Initialize new stack D_r and repeat line 3
18 **end**
19 **foreach** $segs_{ijkm} \in L_1$ **do**
20 **if** $segs_{ijkm}.commit == True$ *or* $segs_{ijkm}.commit == NotValid$ **then**
21 $C_1 \leftarrow C_1 \cup segs_{ijkm}$;
22 $segs_{ijk*}.commit = True$ in SP;
23 **end**
24 **end**
25 $P \leftarrow P \cup C_1$

1. segmentation status, denoted as $seg_{ijkm}.commit$: for all $seg \in SP$, $seg.commit$ will be initialized as $NotValid$; during EDUA-G, if a sentence is first used by a segmentation seg_{ijk}, all segments $seg_{ijk*}.commit$ are set to True, and all other $seg_{ijk'*}$ from this sentence seg_{ij} are set to as False

2. segment status, denoted as $seg_{ijkm}.used$: when initialized, $seg_{ijkm}.used$ is set to False; if segment seg_{ijkm} is used in an SCU, the status $seg_{ijkm}.used$ is set to True

A graph G is constructed from all segments. A list of candidate SCUs (fully connected subgraph) with weights r from n to 2 is exhaustively extracted from G. All the leftover segments with weight as 1 are stored as residuals denote as L_1, at default sorted by the index.

Allocation The allocation process proceeds top-down, iterating over descending values of r from

Algorithm 2: BackTrack

Input: Current set C_r that fails constraints, its size constraint y_r, equivalence class C_{r+1}, current result P, segment pool SP
Result: Adjusted pyramid P with equivalence classes $C_1 \dots C_n$

Initialize $l_{r+1} \leftarrow$C$_{r+1}$, $l_r = \emptyset$;
while $|C_r| \leq y_r$ **do**
  Sort $l_r + 1$ by $\mathcal{AS}$ in ascending order;
  pop e' from $l_r + 1$;
  decompose e' into CU_r where $CU_r = \{CU_{r1}, \dots, CU_{r\binom{r+1}{r}}\}$;
  $l_r \leftarrow l_r \cup CU_r$;
  foreach $e \in l_r$ **do**
    if *notConflict*(C_r, e), $\forall\ seg_{ijkm} \in e$ $seg_{ijkm}.commit == True$ *or* $seg_{ijkm}.commit == NotValid$, *and* $seg_{ijkm}.used == False$ **then**
      Commit(C_r, e, SP) ;
    end
  end
  Update P with new C_r;
  Update C_{r+1} by removing all CU_{r+1}s that have overlapping segments with SCUs in C_r ;
  Update P with new C_{r+1}, reset segment and segmentation status in SP;
  if *P fails to meet any of the constraints* **then**
    BackTrack(C_{r+1}, y_{r+1},C_{r+1+1},P, SP)
  end
  Break ;
end

Algorithm 3: notConflict

Input: current set C_r, a candidate SCU e,
foreach *segment* $seg_{ijkm} \in e$ **do**
  if $\exists\ c \in C_r$ *where* $seg_{ijkm} \in c$ **then**
    return False
  end
end
return True

n to 2. All candidate SCUs are ordered first by weight, then by descending $\mathcal{AS}$. Each set C_r is filled with all candidate SCUs of size r, where maximum $\mathcal{AS}(SCU^r)$ is selected greedily, until the capacity constraint is satisfied. Every SCU committed to C_r requires the segment status to be checked and updated. Then the residual segments are allocated to C_1 as in EDUA-C if the status of the segments permits. For $1 < r < n$, if the provisional pyramid violates any constraints, backtracking considers a provisional revision of C_{r+1} based on reallocating all the segments in each subset of C_{r+1} of size q, for q from 1 to the size of C_{r+1}, considering reallocations in order of descending values of $\mathcal{AP}$. The algorithm terminates when all the constraints are satisfied, no segments remain whose segmentation status is True and whose segment status is False.

Algorithm 4: Commit

Input: current set C_r, a candidate SCU e, segment pool SP
$C_r \leftarrow C_r \cup e$;
foreach $seg_{ijkm} \in e$ **do**
  Set $seg_{ijkm}.used = True$ in SP ;
  Set $seg_{ijkm}.commit = True$ in SP ;
end

Backtracking The backtracking algorithm proceeds bottom-up, from the current set C_r to C_n. Recall from section 4.3, every pair of segments in an SCU has an edge $\geq t_{edge}$; therefore an SCU with $r+1$ contributors can be decomposed into $\binom{r+1}{r}$ SCUs with r contributors. We utilize this property to ensure every set C_r satisfies the constraints. During the emergent search and allocation of SCUs, if a set C_r does not meet the capacity constraint, the backtracking process will be initiated for re-allocation by re-using the segments committed to SCUs in C_{r+1}, to compose new SCUs in C_r. As shown in Algorithm 2, while the allocation process selects SCUs with maximum attraction scores greedily, the backtracking takes a conservative approach of re-doing the commit decision by decomposing one SCU at a time in C_{r+1} with the least $\mathcal{AS}(SCU^r)$, and composing new SCUs with weight r for C_r. It proceeds recursively from r to n until the resulting P satisfies the constraints. This is because every SCU in C_{r+1} has higher importance than in C_r, and this minimizes the impact of the re-allocation step on $\mathcal{AP}$. The backtracking algorithm terminates after all the constraints are satisfied.

B Grid Search on Hyperparameters

Grid search was used to tune the EDUA-G hyperparameters. On DUC 2005 data, we used α in the range [|seg|+10,|seg|+50] where |seg| is the number of input segments, and $\beta \in [1, 3]$. To set t_{edge}, we compute pairwise similarities of all segment pairs from different summaries, and take t_{edge} as the value at percentile N, for $N \in [60, 87]$. The performance metric was correlation with manual pyramid on individual summarization tasks.

Table 5 of ANOVA on the hyperparameters shows that β and t_{edge} have strong impact, while α does not (we select $\alpha = 10$). A contour plot of all combinations of β and t_{edge} (Figure 8) gives two regions of high correlation: $\beta \in [2.5, 3]$, and

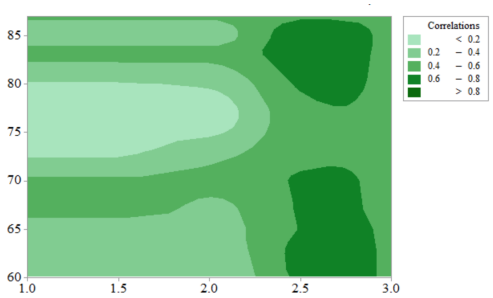


Figure 8: Contour plot for score correlations with β (X-axis) and t_{edge} (Y-axis).

Parameters	**DF**	**F-value**	**P-value**
α	4	0.31	0.872
β	4	104.31	0.000
t_{edge}	11	6.56	0.000

Table 5: One-way ANOVA for hyperparameters, with degrees of freedom (DF), F value and P-value (significance level α=0.05, sample size N=300).

$t_{edge} \in [60, 70]$, or $[80, 87]$. Higher t_{edge} yields fewer edges in the graph, so for efficiency, we select $\beta = 2.5$, and $N = 83$. (Depending on the dataset this corresponds to cosine similarities t_{edge} of about 0.15 to 0.35.)

A Case Study on Combining ASR and Visual Features for Generating Instructional Video Captions

Jack Hessel
Cornell University
jhessel@cs.cornell.edu

Bo Pang **Zhenhai Zhu** **Radu Soricut**
Google
{bopang,zhenhai,rsoricut}@google.com

Abstract

Instructional videos get high-traffic on video sharing platforms, and prior work suggests that providing time-stamped, subtask annotations (e.g., "heat the oil in the pan") improves user experiences. However, current automatic annotation methods based on visual features alone perform only slightly better than constant prediction. Taking cues from prior work, we show that we can improve performance significantly by considering automatic speech recognition (ASR) tokens as input. Furthermore, jointly modeling ASR tokens and visual features results in higher performance compared to training individually on either modality. We find that unstated background information is better explained by visual features, whereas fine-grained distinctions (e.g., "add oil" vs. "add olive oil") are disambiguated more easily via ASR tokens.

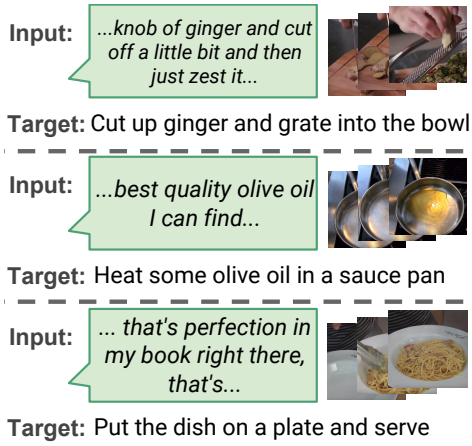


Figure 1: Illustration of a multimodal dense instructional video captioning task. Models are given access to both video frames and ASR tokens, and must generate a recipe instruction step for each video segment. The speaker in the video *sometimes* (but not always) references literal objects and actions.

1 Introduction

Instructional videos increasingly dominate user attention on online video platforms. For example, 86% of YouTube users report using the platform often to learn new things, and 70% of users report using videos to solve problems related to work, school, or hobbies (O'Neil-Hart, 2018).

Prior work in user experience has investigated the best way of presenting instructional videos to users. Kim et al. (2014), for example, compare two options; first: presenting users with the video alone, and second: presenting the video with an additional *structured* representation, including a timeline populated with task subgoals. Users interacting with the structured video representation reported higher satisfaction, and external judges rated the work they completed using the videos as having higher quality. Margulieux et al. (2012) and Weir et al. (2015) similarly find that presenting explicit subgoals alongside how-to videos improves user experiences. Thus, presenting instructional videos with additional structured annotations is likely to benefit users.

These studies rely on human annotation of time-stamped subtask goals, e.g., timed captions created through crowdsourcing. However, human-in-the-loop annotation is infeasible to deploy for popular video sharing platforms like YouTube that receive hundreds of hours of uploads per minute. In this work, we address the task of *automatically* producing captions for instructional videos at the level of video segments. Ideally, generated captions provide a literal, imperative description of the procedural step occurring for a given video segment, e.g., in the cooking context we consider, "add the oil to the pan."

Producing segment-level captions is a sub-task of dense video captioning, where prior work has mostly focused on visual-only models. Dense captioning is a difficult task, particularly in the

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 419–429
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

instructional video domain, as fine-grained distinctions may be difficult or impossible to make with visual features alone. Visual information can be ambiguous (e.g., distinguishing between "olive oil" vs. "vegetable oil") or incomplete (e.g., preparation steps may occur off-camera). In our study, a first important finding is that, for the dataset considered, current state-of-the-art, visual-features–only models only slightly outperform a constant prediction baseline, e.g., by 1.5 BLEU/METEOR points.

To improve performance in this difficult setting, we consider the *automatic speech recognition* (ASR) tokens generated by YouTube. These publicly available tokens are an ASR model's attempts to map words spoken in videos into text. However, while a promising potential source for signal, it is not always trivial to transform even accurate ASR into the desired imperative target: while there are cases of clear correspondence between the literal actions in the video and the ASR tokens, in other cases, the mapping is imperfect (Fig. 1). For example, when finishing a dish, a user says "that's perfection in my book right there" rather than "put the dish on a plate and serve." There are also cases where no ASR tokens are available at all. Despite these potential difficulties, previous work has demonstrated that ASR can be informative in a variety of instructional video understanding tasks (Naim et al., 2014, 2015; Malmaud et al., 2015; Sener et al., 2015; Alayrac et al., 2016; Huang et al., 2017); though less work has focused on instructional caption *generation,* which is known to be difficult and sensitive to input perturbations (Chen et al., 2018).

We find that incorporating ASR-token–based features significantly improves performance over visual-features–only models (e.g., CIDEr improves 0.53 $\Rightarrow$ 1.0, BLEU-4 improves 4.3 $\Rightarrow$ 8.5). We also show that *combining* ASR tokens and visual features results in the highest performing models, suggesting that the modalities contain complementary information.

We conclude by asking: what information is captured by the visual features that *is not* captured by the ASR tokens (and vice versa)? Auxiliary experiments examining performance of models in predicting the presence/absence of individual word types suggest that visual signals are superior for identifying unspoken, implicit aspects of scenes; for instance, in order to mix ingredients, they must be placed in a bowl — and although bowls are often visually present in the scene, "bowl" is often not explicitly mentioned by the speaker. Conversely, ASR features readily disambiguate between fine-grained entities, e.g., "olive oil" vs."vegetable oil", a task that is difficult (and sometimes impossible) for visual features alone.

2 Related Work

Narrated instructional videos. While several works have matched audio and video signals in an unconstrained setting (Arandjelovic and Zisserman, 2017; Tian et al., 2018), our work builds upon previous efforts to utilize accompanying speech signals to understand online *instructional* videos, specifically. Several works focus on learning video-instruction alignments, and match a fixed set of instructions to temporal video segments (Regneri et al., 2013; Naim et al., 2015; Malmaud et al., 2015; Hendricks et al., 2017; Kuehne et al., 2017). Another line of previous work uses speech to extract and align language fragments, e.g., verb-noun pairs, with instructional videos (Gupta and Mooney, 2010; Motwani and Mooney, 2012; Alayrac et al., 2016; Huang et al., 2017, 2018; Hahn et al., 2018). Sener et al. (2015), as part of their parsing pipeline, train a 3-gram language model on segmented ASR token inputs to produce recipe steps.

Dense Video Captioning. Recent work in computer vision addresses dense video captioning (Krishna et al., 2017; Li et al., 2018; Wang et al., 2018), a supervised task that involves (i) segmenting the input video, and, (ii) generating a natural language description for each segment. Here, we focus on the second subtask of generating descriptions given a ground-truth segmentation; this setting isolates the language generation part of the modeling process.[1] Most related to the present work are several dense captioning approaches that have been applied to instructional videos (Zhou et al., 2018b,c). Zhou et al. (2018c) achieve state-of-the-art performance on the dataset we consider; their model is video-only, and combines a region proposal network (Ren et al., 2015) and a Transformer (Vaswani et al., 2017) decoder.

Multimodal Video Captioning. Several works

[1] We find that state-of-the-art models perform poorly even for just this subtask (see § 3.2), so we reserve the full task for future work.

have employed multimodal signals to caption the MSR-VTT dataset (Xu et al., 2016), which consists of 2K video clips from 20 general categories (e.g., "news", "sports") with an average duration of 10 seconds per clip. In particular, Ramanishka et al. (2016); Xu et al. (2017); Hori et al. (2017); Shen et al. (2017); Chuang et al. (2017); Hao et al. (2018) all report small performance gains when incorporating audio features on top of visual features. However — we suspect that instructional video domain is significantly different than MSR-VTT (where the audio information does not necessarily correspond to human speech), as we find that ASR-only models significantly surpass the state-of-the-art video model in our case. Palaskar et al. (2019) and Shi et al. (2019), contemporaneous with the submission of the present work, also examine ASR as a source of signal for generating how-to video captions.

3 Dataset

We focus on YouCook2 (Zhou et al., 2018b), the largest human-captioned dataset of instructional videos publicly available.[2] It contains 2000 YouTube cooking videos, for a total of 176 hours, and spans 89 different recipes. Each video averages at 5.26 minutes, and is annotated with an average of 7.7 temporal segments (i.e., start/end points) corresponding to semantically distinct recipe steps. Each segment is associated with an imperative caption, e.g., "add the oil to the pan", for an average of 8.8 words per caption.

At the time of analysis (June 2018), over 25% of the YouCook2 videos had been removed from YouTube, and therefore we do not consider them. As a result, all our experiments operate on a *subset* of the YouCook2 data. While this makes direct comparison with previous and future work more difficult, our performance metrics can be viewed as lower bounds, as they are trained on less data compared to, e.g., (Zhou et al., 2018c). Unless noted otherwise, our analyses are conducted over 1.4K videos and the 10.6K annotated segments contained therein.

3.1 A Closer Look at ASR tokens

We collected the ASR tokens automatically generated by YouTube (available through the YouTube Data API[3] with trackKind = ASR), which are then mapped to their temporally corresponding video segments. We start by asking the following questions: How much narration do users provide for instructional videos? And: can YouTube's ASR system detect that speech?

Not surprisingly, speakers in videos tend to be more verbose than the annotated groundtruth captions: we find the length distribution of ASR tokens per segment to be roughly log-normal, with mean/median length being 42/28 tokens respectively (compared to a mean of 9 tokens/segment for captions). Over the 10.6K available segments, only 1.6% of them have zero associated tokens. Furthermore, based on automatic language identification provided by the YouTube API and some manual verification, we estimated that less than 1% of videos contain completely non-English speech (but we do not discard them from our experiments).

We also investigate the words-per-minute (WPM) ratio, based on the video segment length. The mean value of 134 WPM is slightly lower than, but comparable to, previously reported figures of English speaking rates (Yuan et al., 2006), which indicates that, for this set of video segments, words are being detected at rates comparable to everyday English speech.

3.2 A Closer Look at the Generation Task

To better understand the generation task, we computed lower and upper bounds for generation performance using a constant-prediction baseline and human performance, respectively.

Lower bound: constant. For all segments at test time, we predict "heat some oil in a pan and add salt and pepper to the pan and stir." This sentence is constructed by examining the most common n-grams in the corpus and pasting them together.

Upper bound: human estimate. We conducted a small-scale experiment to estimate human performance for the segment-level captioning task. Two of the authors of this paper, after being trained on segment-level captions from three videos, attempted to mirror that style of annotation for the segments of 20 randomly sampled videos, totalling over 140 segment annotations each.[4] Both human annotators report low-confidence with the

[2] How2 (Sanabria et al., 2018) tackles the different task of predicting video uploader-provided descriptions/captions, which are not always appropriate summarizations.

[3] https://developers.google.comyoutube/v3/docs/captions

[4] These preliminary experiments are not meant to provide a definitive, exact measure of inter-annotator agreement.

task, in particular, they found it difficult to maintain a consistent level of specificity in terms of how many factual details to include (e.g., "mix together" vs. "mix the peppers and mushrooms together.")

Results: We compute corpus-level performance statistics using four standard generation evaluation metrics: ROUGE-L (Lin, 2004), CIDEr (Vedantam et al., 2015), BLEU-4 (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) (higher is better in all cases).

Note that our evaluation is micro-averaged at the segment level, and differs slightly from prior work on this dataset, which has mostly reported metrics macro-averaged at the video level. We switched the evaluation because some metrics like BLEU-4 exhibit undesirable sparsity artifacts when macro-averaging, e.g., any video without a correct 4-gram gets a zero BLEU score, even if there are many 1/2/3-grams correct. Segment-level averaging, the standard evaluation practice in fields like machine translation, is insensitive to this sparsity concern, and (we believe) provides a more robust perspective on performance.

	BLEU-4	METEOR	ROUGE-L	CIDEr
Constant Prediction	2.70	10.3	21.7	.15
Zhou et al. (2018c)	3.84	11.6	27.4	.38
Sun et al. (2019b)	4.07	11.0	27.5	.50
Sun et al. (2019a)	4.31	11.9	29.5	.53
Human Estimate	15.2	25.9	45.1	3.8

Table 1: The performance of several state-of-the-art, video-only models, with lower (constant prediction) and upper (human estimate) bounds.

This comparison highlights the gap that remains between the simplest possible baseline, several computer vision based models, and (roughly) how well humans perform at this task. Given that Sun et al. (2019a) is a highly tuned computer vision model transfer learned from a corpus of over 300K cooking videos, from the perspective of building video captioning systems in practice, we suspect that incorporating additional modalities like ASR is more likely to result in performance gains versus building better computer vision models.

4 Models

In addition to the constant prediction baseline, we explore a series of ASR-based baseline methods:

ASR as the Caption (ASC) This baseline returns the test-time ASR token sequence as the caption. While the result is not a coherent, imperative step, performance of this method offers insight into the extent of word overlap between the ASR sequence and the target groundtruth, as measured by the captioning metrics.

Filtered ASR (FASC) Given that the ASR token sequences are much longer than groundtruth captions (§ 3.1), the performance of ASC incurs a length (or precision-based) penalty for several metrics. The FASC baseline strengthens ASC by removing word types that are less likely to appear in groundtruth captions, e.g., "ah", "he", "hello," or "wish". Specifically, we only keep words with high $\frac{P(w \mid GT)}{P(w \mid ASR)}$ values, i.e., words that would be indicative of the groundtruth class if we were to build a Naive-Bayes classifier with add-one smoothing; probabilities are computed only over the training set to reduce the risk of overfitting. This baseline produces outputs that are shorter compared to ASC, but it is unlikely to yield fluent, readable text.

ASR-based Retrieval (RET) This retrieval baseline memorizes the recipe steps in the training set, and represents them each as tf-idf vectors. At test-time, the ASR sequence is converted into a tf-idf vector and compared to each training-set caption via cosine similarity.[5] The training caption that is most similar to the test-time ASR according to this metric is returned as the "generated" caption. Note that, although a memorization-based technique, this baseline method produces de-facto captions as outputs.

4.1 Transformer-based Neural Models

We explore neural encoder-decoder models based on Transformer Networks (Vaswani et al., 2017). In contrast to RNNs, Transformers abandon recurrence in favor of a mix of different types of feed-forward layers, e.g., in the case of the Transformer decoder, self-attention layers, cross-attention layers (attending to the encoder outputs), and fully connected feed-forward layers. We explore two variants of the Transformer, corresponding to different hypotheses about what information might be useful for captioning instructional videos.

ASR Transformer (AT) This model learns to map ASR-token sequences directly to captions using

[5] We tried several variants of this method, e.g., comparing test ASR to train ASR, but found that comparing test ASR to train captions performed the best.

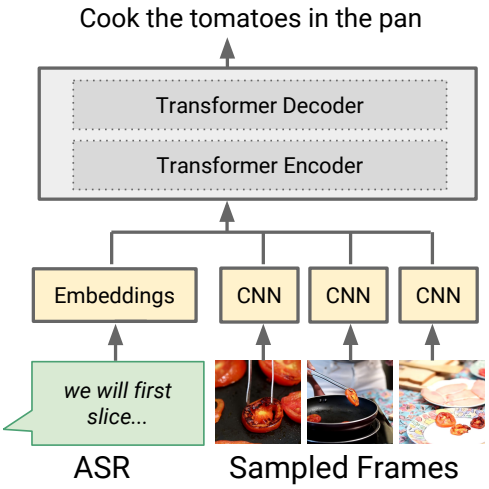


Figure 2: The AT+Video model. Both the encoder and decoder layers perform cross-modal attention.

	BLEU-4	METEOR	ROUGE-L	CIDEr
CNST	2.70	10.03	21.69	0.15
Sun et al. (2019a)	4.31	11.91	29.47	0.53
ASC	1.68	14.86	19.24	0.20
FASC	4.32	18.47	30.07	0.59
RET	5.68	14.29	28.06	0.80
AT	8.55	16.93	35.54	1.06
AT+Video	9.01	17.77	**36.65**	**1.12**

Table 2: Caption generation performance: AT+Video is a multimodal model that adds visual frame features to AT. A bolded value in a column indicates a statistically-significant improvement, whereas an underline indicates a statistical tie for best ($p < .01$).

a standard sequence-to-sequence Transformer architecture. The model's parameters are optimized to maximize the probability of the ground-truth instructions, conditioned on the input ASR sequences.

Multimodal model (AT+Video) We incorporate video features into the ASR transformer (Fig 2). For ease of comparison with prior and future work, we use features extracted from ResNet34 (He et al., 2016) pretrained on the ImageNet classification task; these features are provided in the YouCook2 data release. Each video is initially uniformly sampled at 512 frames, with an average of 30 frames per captioned-segment.

To represent each video segment, first, k frames are randomly sampled with replacement. The sampled frames are temporally sorted to preserve ordering information, and their corresponding ResNet34 feature vectors are projected to the Transformer encoder hidden dimension via a width-1 1D convolution. We use $k = 10$ for all our experiments. The encoder self-attention layers perform *cross-modal attention* operations between the visual features and the ASR-token–based features. For each output token, the decoder attends to previously predicted tokens, and encoder outputs for all input frames / ASR tokens.

5 Experiments

We perform 10-fold cross-validation with randomly sampled 80/10/10 train/dev/test splits (split at the video-level), using the same splits for all models. After discarding the videos that were deleted at the time of data collection, each split contains roughly 1.1K training videos (averaging 8.3K training segments). We report mean performance over these splits according to four standard captioning accuracy metrics, introduced in §3.2. ROUGE-L, CIDEr, BLEU-4, and METEOR. We perform both Wilcoxon signed-rank tests (Demšar, 2006) and two-sided corrected resampled t-tests (Nadeau and Bengio, 2000) to estimate statistical significance. To be conservative and reduce the chance of Type I error, we take whichever p-value is larger between these two tests.

Transformer-based model details. For each cross-validation split, we use a batch size of 128, tie the Transformer model's feed forward and model dimensions $d_{ffn} = d_{model}$, and optimize regularized cross-entropy loss using Adam (Kingma and Ba, 2015) with $lr = .001$. We train models for 100K steps, storing checkpoint files periodically. For each split, we train 8 model variants, conducting a grid search over model dimension, number of encoder/decoder layers, and L2 regularization: we consider all model parameter settings in $(d_{model}, N_{layer}, \lambda_{reg}) \in \{128, 256\} \times \{2, 3\} \times \{.0005, .001\}$ for each cross-validation split independently, and select the highest performing, checkpointed model according to ROUGE-L over the development set for that fold. Transformer models are implemented using `tensor2tensor` (Vaswani et al., 2018) and `Tensorflow` (Abadi et al., 2015). The vocabulary (average size 800) is determined separately using the training data for each cross-validation split. Words are considered if they occur at least 5 times in the ground-truth of the current training set.[6] This leads to an OOV rate of ~60% in the input. We truncate inputs at 80 tokens (~10-15%

[6]Different vocabulary creation schemes, e.g., sub-word tokenization, led to small performance decreases.

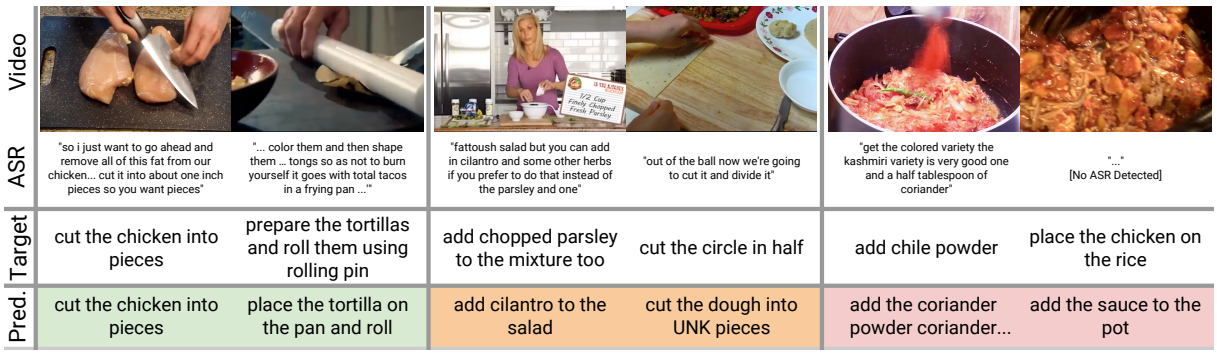


Figure 3: Example generations from AT+Video in cases where it performs **well**, **okay**, and **poorly**.

of transcripts are truncated in this process). For simplicity, decoding is done greedily in all cases.

Generation Experiment Results. Table 2 reports the performance of each model. For unimodal models, simple baselines like FASC (filtered ASR) and RET (training-caption retrieval) outperform the state-of-the-art video-only model of Sun et al. (2019a), according to the four automatic evaluation metrics. Overall, AT yields the best unimodal performance. Combining ASR and visual signals into a multimodal representation performs even better: the AT+Video model tends to outperform AT (and Sun et al. (2019a)), according to ROUGE-L, CIDEr, and METEOR ($p < .01$). Since AT and AT+Video have identical architectures and differ only in the available inputs, this result provides strong evidence that it is indeed the *multimodality* of AT+Video that leads to the (statistically significant) performance gains over the strongest unimodal models. We present some output examples in Fig. 3.

5.1 Diversity of Generated Captions

In addition to the automatic quality metrics, we measure how diverse the generated caption are for each model, using the following metrics: vocabulary coverage (the percent of vocabulary that was predicted at test-time by each algorithm at least once); proportion not copied (the percent of generated captions that do not appear in the training set verbatim); and output uniqueness (the percent of generated captions that are unique). These metrics are useful because they can highlight undesirable, degenerate behavior for models.[7] As an upper-bound, we compute these metrics for the ground-truth (GT) test-time targets. Note that even the ground-truth targets do not achieve 100% in these diversity metrics: for vocabulary coverage, not all vocabulary items appear in the ground-truth captions for a given cross-validation split; similarly, for proportion not copied/output uniqueness, because there are repeated captions in the label set.

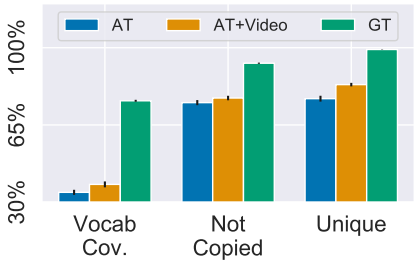


Figure 4: The multimodal model AT+Video produces slightly more diverse captions than its unimodal counterparts.

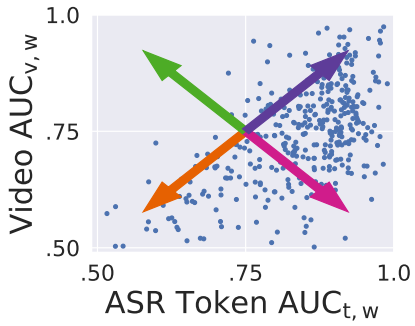
According to all metrics, AT+Video outputs are slightly more diverse compared to the AT outputs (Fig. 4). This observation suggests that the multimodal model is not simply exploiting a degeneracy to achieve its performance improvements.

6 Complementarity of Video and ASR

We now turn to the question of *why* multimodal models produce better captions: what type of signal does video contain that speech does not (and vice versa)? Our initial idea was to quantitatively compare the captions generated by AT versus AT+Video; however, because the dataset is relatively small, we were unable to make observations about the generated captions that were statistically significant.[8]

[7] For instance, the constant prediction baseline we consider would score low in both vocab coverage and uniqueness.

[8] In general, making concrete statements about the causal link between inputs and outputs of sequence-to-sequence models is challenging, even in the text-to-text case, see Alvarez-Melis and Jaakkola (2017).



knead	97.8
nori	97.1
yeast	96.1
mozzarella	95.8
lettuce	95.3
pancake	94.7
wrapper	94.3
patty	93.4
dal	93.0
grill	92.9
pizza	92.9
oven	92.7
bake	92.3

(a) **Easiest** (AUC$_{\mu,w}$)

4	43.8
bit	51.7
about	52.1
prepare	52.3
mixed	54.5
then	54.5
spoon	54.9
or	55.9
it	56.2
ready	56.7
3	57.1
few	58.3
more	58.5

(b) **Hardest** (AUC$_{\mu,w}$)

fat	39.7
turn	36.4
sea	35.9
white	31.7
chilies	30.8
dried	30.6
beer	30.3
pancetta	30.0
mustard	29.8
spice	28.4
sliced	28.3
cinnamon	28.0
warm	27.8

(c) **ASR Better** (AUC$_{\Delta,w}$)

sandwich	-18.0
stove	-15.4
tuna	-14.3
again	-12.6
mince	-11.2
wok	-8.9
burger	-8.8
pizza	-8.4
serve	-7.8
4	-6.6
mussels	-6.6
tray	-6.3
bowl	-5.9

(d) **Video Better** (AUC$_{\Delta,w}$)

Figure 5: Per-word classification results using ASR and/or Video features. Each point in the scatterplot represents a different word-type; x-coordinate values show how well a word is predicted by ASR-token features; y-coordinate values show how well a word is predicted by video features. Tables (a)-(d) show word types that are easy, universally difficult, better-predicted-by-ASR, and better-predicted-by-video, respectively.

Instead, we examine properties of the ASR-token–based and visual features directly. Following a procedure inspired from (Lu et al., 2008; Berg et al., 2012; Dai et al., 2018; Mahajan et al., 2018), we consider the auxiliary task of predicting presence/absence of unigrams in the ground truth captions from features extracted from corresponding segments. We train two unimodal classifiers, one using ASR-token–based features and one using visual features, and measure their relative capacity to predict different word types; the goal is to measure which word types are most-predictable from the ASR tokens and, conversely, which ones are most-predictable from the visual features.

For each segment, we predict the unigram distribution of its corresponding caption using a unimodal softmax classifier: for simplicity, we use a 2-layer, residual deep averaging network (Iyyer et al., 2015) for both the visual and ASR-based classifier. We measure per-word-type performance using AUC, which is word-frequency independent.

Specifically — for each word type w (e.g., w = `beer`) we measure how well w is predicted by the classifier based on ASR / spoken tokens AUC$_{t,w}$ (e.g., AUC$_{t,beer}$ = 98) and, conversely, how well w is predicted by the visual classifier AUC$_{v,w}$ (AUC$_{v,beer}$ = 68). For a given word type, we measure its overall difficulty by averaging AUC$_{t,w}$ and AUC$_{v,w}$; we call this AUC$_{\mu,w}$ (AUC$_{\mu,beer}$ = 83). Similarly, we measure the difference in difficulty by subtracting AUC$_{t,w}$ and AUC$_{v,w}$ to give AUC$_{\Delta,w}$ (AUC$_{\Delta,beer}$ = 30) with higher values indicating that a word type is predicted better by the spoken-token features compared to the visual features. We plot AUC$_{t,w}$ versus AUC$_{v,w}$ for 382 words in Fig. 5 (results are averaged over 10 cross-val splits).

Absolute Performance. Points in the upper-right quadrant of Fig. 5 represent words that are easy for both visual and ASR-token–based features to predict, whereas points in the lower-left represent words that are more difficult. Specific ingredients, e.g., "nori" and "mozzarella," are often easy to detect, as are actions closely associated with particular objects (e.g., "dough" is almost always the object being "knead"-ed). Conversely, pronouns (e.g., "it") and conjunctions (e.g., "or") are universally difficult to predict.

Visual vs. ASR-token–based features. In general, ASR-token–based features carry greater predictive power, as evidenced by the skew towards the bottom right in the scatterplot in Fig. 5. One pattern in the cases where speech features perform better (Fig. 5c) is that words are often modifiers, e.g., *white* (pepper), *sea* (salt), *dried* (chilies), *olive* (oil), etc. Indeed, small, detailed distinctions may be often difficult to make from visual features, e.g., "vegetable oil" and "olive oil" may look identical in most YouTube videos.

Nonetheless, there are types better predicted by video features (Fig. 5d). Often, these are cases that require unstated, background knowledge, i.e., references to objects not explicitly stated by the speaker(s). To quantify this observation, for each word type we compute the likelihood that it is *stated* by the speaker in the video, given that it appears in the ground-truth caption, i.e., $P(w \in \texttt{ASR} \mid w \in \texttt{GT})$. Aside from trivial cases (e.g., words misspelled in the GT never appear in the ASR), words that are often unstated include action words (e.g., "place", "crush") and cookware (e.g., "pan", "wok", "pot"). Words that are often stated include specific ingredients (e.g., "honey", "coconut", "ginger"). In contrast to word frequency (which is uncorrelated with AUC$_{\Delta,w}$, Spearman

$\rho \approx 0$), stated rate *is* correlated with $\text{AUC}_{\Delta,w}$ ($\rho = 0.44, p < .01$).

7 Oracle Object Detection

The results in Table 2 indicate that, while adding visual information yields statistically significant improvements to the ASR-only model, the improvements are not large in magnitude. This leaves open the question of whether (a) any visual information simply does not provide much additional information on top of ASR, or (b) we need better visual modeling. We take a first step in addressing this question by experimenting with an "oracle" object detector that provides perfect-precision predictions.[9] If even oracle object detection does not help, then the answer is more likely (a) rather than (b) above.

As part of a YouCook2 data release, bounding box annotations for selected objects in the recipe text (Zhou et al., 2018a) were provided. Unfortunately, while these could have served as an oracle, the actual annotations are only available for a small fraction of the data. Instead, we consider the set of 62 object labels made available. We simulate a high-precision, oracle object detector by identifying – per video segment – the overlap between (morphology-normalized) groundtruth caption mentions and the 62 object labels available.[10] For instance, for the groundtruth caption "put the mushrooms in the pan", the oracle object detector yields "mushroom" and "pan". 89% of segments receive at least one oracle object. The oracle object detections are then fed into the Transformer encoder (in random order), either by themselves (Oracle) or along with the ASR token sequence (AT+Oracle). We perform the same cross-validation experiments as described in §5, and report the average ROUGE-L (we observe similar trends with other metrics):

	AT	AT+Video	Oracle	AT+Oracle
ROUGE-L	35.5	36.7	40.8	**45.5**

Because the AT+Oracle model achieves large improvements over AT+Video, we suspect that building higher-quality visual representations is a promising avenue for future work.

[9] High-precision object detectors are gaining popularity in the computer vision community because the training data is easier to annotate, e.g., Krasin et al. (2017).

[10] This oracle is unlikely to be achievable, as it assumes 100% precision for the 62 objects considered (which also implies modeling *which* objects to talk about, a non-trivial task in itself (Berg et al., 2012)).

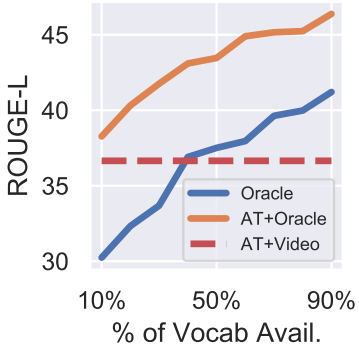


Figure 6: The performance of the oracle methods increases as they are given access to an increasing number of object types.

How weak of an oracle can still produce high performance? Fig. 6 shows performances of models using *subsets* of the 62 objects (most frequent 10% of objects through 90%) over one cross-validation fold. AT+Oracle gives better performance than AT+Video by detecting *just 6 object types,* and the oracle by-itself (which is only given access to object sets) achieves comparable performance to AT+Video with 30 object types. These results suggest that, at least for this task, the Transformer decoder is likely not the main performance bottleneck, as it is able to paste-together unordered object detections into captions effectively.

8 Conclusion

In this work, we demonstrate the impact of incorporating both visual and ASR-token–based features into instructional video captioning models. Additional experiments investigate the complementarity of the visual and speech signals.

Our oracle experiments suggest that performance bottlenecks likely derive from the input encoding, as the decoder is able to paste-together even simple sets of object detections into high-quality captions. Future work would thus be well-suited to investigate better models of input data. Given the small size of the dataset, transfer learning may prove fruitful, e.g., pre-training the encoder with an unsupervised, auxiliary task; work contemporaneous with our submission from the computer vision community suggests that transfer learning indeed is a promising direction (Sun et al., 2019b,a; Miech et al., 2019).

Acknowledgements. We would like to thank Maria Antoniak, Nan Ding, Sebastian Goodman, Jean Griffin, Fernando Pereira, Chen Sun, and the anonymous reviewers for their helpful comments.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien. 2016. Unsupervised learning from narrated instruction videos. In *CVPR*.

David Alvarez-Melis and Tommi S Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. In *EMNLP*.

Relja Arandjelovic and Andrew Zisserman. 2017. Look, listen and learn. In *ICCV*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL workshop on Evaluation Measures for MT and Summarization*.

Alexander C Berg, Tamara L Berg, Hal Daumé III, Jesse Dodge, Amit Goyal, Xufeng Han, Alyssa Mensch, Margaret Mitchell, Aneesh Sood, Karl Stratos, et al. 2012. Understanding and predicting importance in images. In *CVPR*.

Hongge Chen, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, and Cho-Jui Hsieh. 2018. Attacking visual language grounding with adversarial examples: A case study on neural image captioning. In *ACL*.

Shun-Po Chuang, Chia-Hung Wan, Pang-Chi Huang, Chi-Yu Yang, and Hung-Yi Lee. 2017. Seeing and hearing too: Audio representation for video captioning. In *IEEE Automatic Speech Recognition and Understanding Workshop*.

Bo Dai, Sanja Fidler, and Dahua Lin. 2018. A neural compositional paradigm for image captioning. In *NeurIPS*.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *JMLR*.

Sonal Gupta and Raymond J Mooney. 2010. Using closed captions as supervision for video activity recognition. In *AAAI*.

Meera Hahn, Nataniel Ruiz, Jean-Baptiste Alayrac, Ivan Laptev, and James M Rehg. 2018. Learning to localize and align fine-grained actions to sparse instructions. *arXiv preprint arXiv:1809.08381*.

Wangli Hao, Zhaoxiang Zhang, and He Guan. 2018. Integrating both visual and audio cues for enhanced video caption. In *AAAI*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *ICCV*.

Chiori Hori, Takaaki Hori, Teng-Yok Lee, Ziming Zhang, Bret Harsham, John R Hershey, Tim K Marks, and Kazuhiko Sumi. 2017. Attention-based multimodal fusion for video description. In *ICCV*.

De-An Huang, Shyamal Buch, Lucio Dery, Animesh Garg, Li Fei-Fei, and Juan Carlos Niebles. 2018. Finding it: Weakly-supervised reference-aware visual grounding in instructional videos. In *CVPR*.

De-An Huang, Joseph J Lim, Li Fei-Fei, and Juan Carlos Niebles. 2017. Unsupervised visual-linguistic reference resolution in instructional videos. In *CVPR*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *ACL*.

Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J Guo, Robert C Miller, and Krzysztof Z Gajos. 2014. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *CHI*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. 2017. Openimages: A public dataset for large-scale multi-label and multi-class image classification.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. 2017. Dense-captioning events in videos. In *ICCV*.

Hilde Kuehne, Alexander Richard, and Juergen Gall. 2017. Weakly supervised learning of actions from transcripts. *Computer Vision and Image Understanding*.

Yehao Li, Ting Yao, Yingwei Pan, Hongyang Chao, and Tao Mei. 2018. Jointly localizing and describing events for dense video captioning. In *CVPR*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Yijuan Lu, Lei Zhang, Qi Tian, and Wei-Ying Ma. 2008. What are the high-level concepts with small semantic gaps? In *CVPR*.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the limits of supervised pretraining. In *ECCV*.

Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. 2015. What's cookin'? interpreting cooking videos using text, speech and vision. In *NAACL*.

Lauren E Margulieux, Mark Guzdial, and Richard Catrambone. 2012. Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. In *Conference on International Computing Education Research*.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *ICCV*.

Tanvi S Motwani and Raymond J Mooney. 2012. Improving video activity recognition using object recognition and text mining. In *ECAI*.

Claude Nadeau and Yoshua Bengio. 2000. Inference for the generalization error. In *NeurIPS*.

Iftekhar Naim, Young C Song, Qiguang Liu, Liang Huang, Henry Kautz, Jiebo Luo, and Daniel Gildea. 2015. Discriminative unsupervised alignment of natural language instructions with corresponding video segments. In *NAACL*.

Iftekhar Naim, Young Chol Song, Qiguang Liu, Henry A Kautz, Jiebo Luo, and Daniel Gildea. 2014. Unsupervised alignment of natural language instructions with video segments. In *AAAI*.

Celie O'Neil-Hart. 2018. Why you should lean into how-to content in 2018. www.thinkwithgoogle.com/advertising-channels/video/self-directed-learning-youtube/. Accessed: 2019-09-03.

Shruti Palaskar, Jindrich Libovickỳ, Spandana Gella, and Florian Metze. 2019. Multimodal abstractive summarization for how2 videos. In *ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Vasili Ramanishka, Abir Das, Dong Huk Park, Subhashini Venugopalan, Lisa Anne Hendricks, Marcus Rohrbach, and Kate Saenko. 2016. Multimodal video description. In *ACM MM*.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *TACL*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*.

Ramon Sanabria, Ozan Caglayan, Shruti Palaskar, Desmond Elliott, Loïc Barrault, Lucia Specia, and Florian Metze. 2018. How2: a large-scale dataset for multimodal language understanding. In *Proceedings of the Workshop on Visually Grounded Interaction and Language (ViGIL)*. NeurIPS.

Ozan Sener, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. 2015. Unsupervised semantic parsing of video collections. In *ICCV*.

Zhiqiang Shen, Jianguo Li, Zhou Su, Minjun Li, Yurong Chen, Yu-Gang Jiang, and Xiangyang Xue. 2017. Weakly supervised dense video captioning. In *CVPR*.

Botian Shi, Lei Ji, Yaobo Liang, Nan Duan, Peng Chen, Zhendong Niu, and Ming Zhou. 2019. Dense procedure captioning in narrated instructional videos. In *ACL*.

Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. 2019a. Contrastive bidirectional transformer for temporal representation learning. *arXiv preprint arXiv:1906.05743*.

Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019b. Videobert: A joint model for video and language representation learning. In *ICCV*.

Yapeng Tian, Jing Shi, Bochen Li, Zhiyao Duan, and Chenliang Xu. 2018. Audio-visual event localization in unconstrained videos. In *ECCV*.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. *CoRR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*.

Jingwen Wang, Wenhao Jiang, Lin Ma, Wei Liu, and Yong Xu. 2018. Bidirectional attentive fusion with context gating for dense video captioning. In *CVPR*.

Sarah Weir, Juho Kim, Krzysztof Z Gajos, and Robert C Miller. 2015. Learnersourcing subgoal labels for how-to videos. In *CSCW*.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. MSR-VTT: A large video description dataset for bridging video and language. In *CVPR*.

Jun Xu, Ting Yao, Yongdong Zhang, and Tao Mei. 2017. Learning multimodal attention lstm networks for video captioning. In *ACM MM*.

Jiahong Yuan, Mark Liberman, and Christopher Cieri. 2006. Towards an integrated understanding of speaking rate in conversation. In *International Conference on Spoken Language Processing*.

Luowei Zhou, Nathan Louis, and Jason J Corso. 2018a. Weakly-supervised video object grounding from text by loss weighting and object interaction. In *BMVC*.

Luowei Zhou, Chenliang Xu, and Jason J Corso. 2018b. Towards automatic learning of procedures from web instructional videos. In *AAAI*.

Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. 2018c. End-to-end dense video captioning with masked transformer. In *CVPR*.

Leveraging Past References for Robust Language Grounding

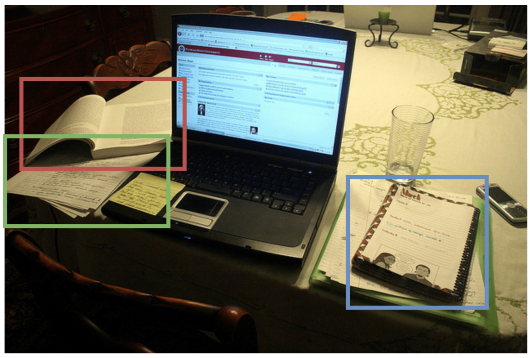
Subhro Roy[*], **Michael Noseworthy**[*], **Rohan Paul, Daehyung Park, Nicholas Roy**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{subhro, mnosew, rohanp, daehyung, nickroy}@csail.mit.edu

Abstract

Grounding referring expressions to objects in an environment has traditionally been considered a one-off, ahistorical task. However, in realistic applications of grounding, multiple users will repeatedly refer to the same set of objects. As a result, past referring expressions for objects can provide strong signals for grounding subsequent referring expressions. We therefore reframe the grounding problem from the perspective of coreference detection and propose a neural network that detects when two expressions are referring to the same object. The network combines information from vision and past referring expressions to resolve which object is being referred to. Our experiments show that detecting referring expression coreference is an effective way to ground objects described by subtle visual properties, which standard visual grounding models have difficulty capturing. We also show the ability to detect object coreference allows the grounding model to perform well even when it encounters object categories not seen in the training data.

1 Introduction

Grounding referring expressions to objects in an environment is a key *Artificial Intelligence* challenge spanning computer vision and natural language processing. Past work in referring expression grounding has focused on understanding the ways a human might resolve ambiguity that arises when multiple similar objects are in a scene – for example, by referring to visual properties or spatial relations (Nagaraja et al., 2016; Hu et al., 2017). However, most previous work treats it as an *ahistorical* task: a user is presented with an image and a referring expression and only features from the current referring expression are used to determine which object is being referred to. However, in many scenarios where a grounding system can be deployed, grounding is not an isolated one-off task. Instead, users will repeatedly refer to the same set of objects. In a household environment, a robot equipped with a grounding system will repeatedly be asked to retrieve objects by the people who live there. Similarly, during a cooperative assembly task, a robot will repeatedly be asked for various parts or tools. Even in non-embodied systems, such as conversational agents, a user will repeatedly refer to different relevant entities.



Utterances from a User:
(1) The open text book.
(2) Page with a dark border and images of two people at the bottom.
(3) The open book with typed pages.
(4) Pages of handwritten notes under the open book.

Figure 1: An example where a single user repeatedly refers to objects in the same scene. The color identifies which bounding box is being referred to.

These scenarios present new challenges and opportunities for grounding referential expressions. When people in the same environment commonly interact with each other (as in a household or workplace), *lexical entrainment* will likely occur (Brennan and Clark, 1996). That is, users of the system will come to refer to objects in similar ways to how they have been referred to in the past.

[*] Equal contribution.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 430–440
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

Thus, it is important that the grounding system can adapt to the vocabulary used where it is deployed.

Even if each object is being repeatedly referred to by a set of people who do not interact with each other, the agent can still learn general information about the objects by remembering how they have been referred to in the past. This is useful as it provides a way for the model to learn properties of objects that are otherwise hard to detect such as subtle visual properties which cannot easily be captured by standard visual grounding systems.

To include past referring expressions in a grounding model, we formulate grounding as a type of coreference resolution – are a new phrase and a past phrase (known to identify a specific object) referring to the same object? To compare a new referring expression with past referring expressions, we need to learn a type of compatibility metric that tells us whether two expressions can both describe the same object. Detecting object coreference involves distinguishing between mutually exclusive object attributes, recognizing taxonomic relations, and permitting unrelated but possibly co-existing properties.

Our contribution is to demonstrate that grounding accuracy can be improved by incorporating a module that has been trained to perform coreference resolution to previous referring expressions. We introduce a neural network module that learns to identify when two referring expressions describe the same object. By jointly training the system with a visual grounding module, we show how grounding can be improved using information from both linguistic and visual modalities.

We evaluate our model on a dataset where users repeatedly refer to objects in the same scene (see Figure 1). Given the same amount of training data, our coreference grounding model achieves an overall increase of 15% grounding accuracy when compared to a state-of-the-art visual grounding model (Hu et al., 2017). We show that the coreference grounding model can better generalize to object categories and their descriptions not seen during training – a common difficulty of visual grounding models. Finally, we show that jointly training the coreference model with a visual grounding model allows the joint model to use object properties not stated in previous referring expressions. As an example application, we demonstrate how the coreference grounding paradigm can be used with a robotic platform.[1]

2 Technical Approach

The task of grounding referring expressions is to identify which object is being described by a query referring expression, Q. The input to this problem is a set of N objects, $O = \{o_1, o_2, \ldots, o_N\}$.[2] Each object, o_i, is represented by its visual features, v_i (i.e., pixels from the object's bounding box), and a referring expression, r_i, that was previously used to refer to that object (r_i may not always be available). The grounding problem can then be modelled as estimating the distribution over which object is being referred to: $p(x|Q, v_{1:N}, r_{1:N})$ where x is a random variable with domain O.

There has been a lot of work in grounding referring expressions (Hu et al., 2017), many of which use only visual features and no interaction history. Most of the proposed models have the form:

$$p(x = o_i|Q, v_{1:N}) = \mathcal{S}\left(\mathrm{W}_{vis} \cdot f_{vis}\left(Q, v_{1:N}\right)\right)_i = \frac{\exp\left(\mathrm{W}_{vis} \cdot f_{vis}\left(Q, v_i\right)\right)}{\sum_{j=1}^{N} \exp\left(\mathrm{W}_{vis} \cdot f_{vis}\left(Q, v_j\right)\right)}$$

where $f_{vis}\left(Q, v_i\right)$ is a low dimensional representation of the visual features v_i and the query Q, W_{vis} is a learned linear transformation, and $\mathcal{S}(\cdot)_i$ is the ith entry of the softmax function output.

We introduce a similar model for coreference grounding which uses past referring expressions to decide which object o_i is being referred to:

$$p\left(x = o_i|Q, r_{1:N}\right) = \mathcal{S}\left(\mathrm{W}_{coref} \cdot f_{coref}\left(Q, r_{1:N}\right)\right)_i$$

where $f_{coref}\left(Q, r_i\right)$ is an embedding of a past referring expression and the query expression, and W_{coref} is a learned linear transformation.

Finally, we introduce a joint model which fuses representations f_{vis} and f_{coref} by a function g:

$$p\left(x = o_i|Q, v_{1:N}, r_{1:N}\right) = \mathcal{S}\left(g\left(f_{coref}\left(Q, r_{1:N}\right), f_{vis}\left(Q, v_{1:N}\right)\right)\right)_i \quad (1)$$

Note that f_{vis} can come from any visual grounding model that associates text to visual features extracted from the objects' bounding boxes.

[1] The dataset, code, and demonstration videos can be found at https://mike-n-7.github.io/coreference-grounding.html.

[2] Object proposal networks (e.g., Faster R-CNN) can be used to extract object bounding boxes from an image.

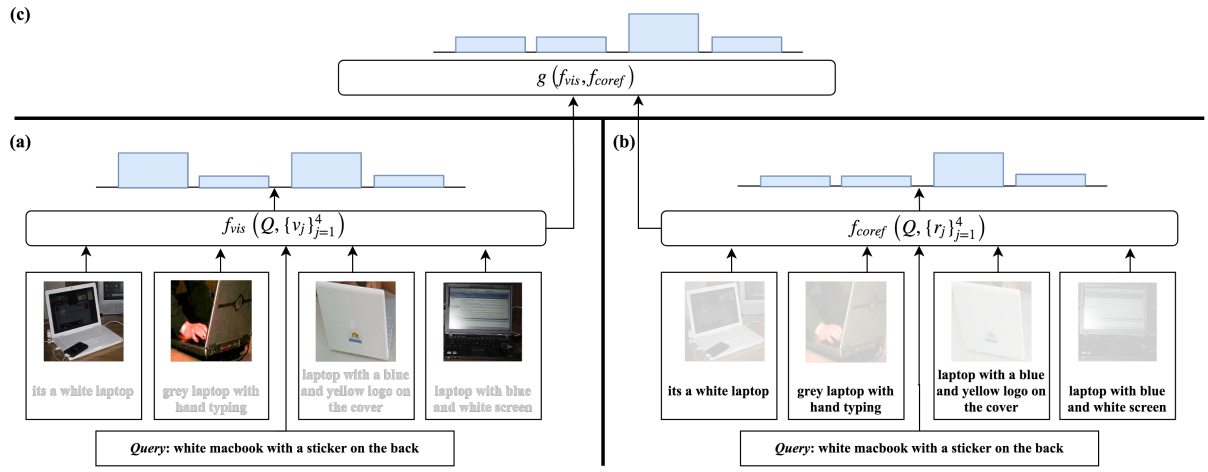


Figure 2: (a) A visual grounding model only takes in images to resolve the query expression and outputs a categorical distribution over the objects. (b) We propose a model that uses past referring expressions to resolve the new expression. (c) These models can be combined to fuse visual and linguistic information.

Our contributions are the coreference and joint grounding models. Both models learn to ground a new referring expression by computing compatibility with past referring expressions of candidate objects. We describe f_{coref} in detail in Section 2.1 and the joint model in Section 2.2. For a visualization of the model, see Figure 2.

2.1 Coreference Grounding Model

Given a query referring expression Q and an expression r_i for each object (each represented by a sequence of M words: $\{w_1, w_2, \ldots, w_M\}$), we define a joint representation of these two expressions, f_{coref}, as follows:

$$f_{coref}(Q, r_i) = f_{enc}(Q) \odot f_{enc}(r_i)$$

where f_{enc} produces referring expression embeddings for the given phrase of dimension $l \times 1$, and $\odot$ is the elementwise multiplication operator. Note the same encoder is used to embed both r_i and Q. In Section 4.3, we evaluate various referring expression embedding methods described below.

LSTM Embeddings The final output state of an LSTM (Hochreiter and Schmidhuber, 1997) is used as the referring expression embedding.

BiLSTM Embeddings For the bidirectional LSTM, forward and backward LSTMs are run over the input sequence, and their outputs for each word are concatenated. An expression embedding is computed by the dimension-wise max across words (Collobert and Weston, 2008).

Attention Embeddings Attention encoders (Lin et al., 2017) learn a weighted average of BiLSTM outputs as the referring expression representation. They output an attention score that is used to compute a weighted average of the BiLSTM outputs.

InferSent Embeddings Recently, *InferSent* (Conneau et al., 2017) was proposed as a *general purpose* sentence embedding method. *InferSent* is similar to the BiLSTM model but was trained using the *Natural Language Inference* task with the intuition that this task would require the sentence embeddings to contain semantically meaningful information. The authors showed that their sentence representation generalized well to other tasks. The pretrained encoder from the *InferSent* model is used to embed a referring expression.

2.2 Integrating Vision and Coreference

The coreference model can learn the complexities of coreference with past expressions, but if certain properties are not mentioned in a previous expression, the model will have difficulty deciding which object is being referred to. If the referenced property could have been learned by a visual grounding model, we would like to include this representation in our model. In this section, we show how we can take an existing visual grounding model and combine it with a coreference grounding model. We generate representations f_{vis} from an existing grounding model, and f_{coref} from our coreference grounding model. These representations are fused using a function $g(\cdot)$, which is used to compute the most likely referred object using Equation 1. We experiment with two choices of g, which we describe in the following subsections.

2.2.1 Addition

One approach is to take the sum of the scores of component visual and coreference grounding models. The function $g(\cdot)$ can be written as (arguments of functions removed for brevity):

$$g = \mathrm{W}_{vis} \cdot f_{vis}(\cdot) + \mathrm{W}_{coref} \cdot f_{coref}(\cdot)$$

where $\mathrm{W}_{vis}, \mathrm{W}_{vis} \in \mathbb{R}^{1 \times l}$ are learned parameters which transform their respective feature vectors of size l into scalar scores.

2.2.2 Concatenation

Simple addition might not capture all interdependencies between different modalities. We propose an approach where we concatenate the representations f_{vis} and f_{coref}, and add a two layer feed forward network to output the final score,

$$g = \mathrm{W}_{c1}(\mathrm{ReLU}(\mathrm{W}_{c2}(\mathrm{ReLU}([f_{vis} f_{coref}]).$$

For our visual grounding model, we use the *Compositional Modular Network* (Hu et al., 2017), which is one of the top performing models in several benchmark referring expressions datasets. We train the joint model end-to-end so that it can learn how to properly merge the visual and coreference information.

3 Dataset

Our goal is to ground a referring expression to an object in the environment using past referring expressions and visual features. Existing referring expression datasets do not contain at least two referring expressions for each object. Since this is a requirement to learn and evaluate models that can utilize past expressions, we create two new datasets for the task – a large *Diagnostic* dataset where past expressions are algorithmically assigned, and a smaller *Episodic* dataset created to capture more realistic interaction scenarios.

3.1 Diagnostic Dataset

In this dataset, artificial scenes are created by grouping similar objects, and each referring expression for an object is collected independently. This form of dataset allows us to easily scale up the amount of data used for training and capture more descriptive language by introducing category-level ambiguity into the scene. We use images from the MSCOCO dataset (Lin et al., 2014), which contains bounding boxes for each object in an image. We randomly group together four object images from the same category. We randomly label one object as the goal object and the remaining three as distractor objects. Annotators from the *Figure Eight* platform[3] are shown these images with the goal object labeled by a red bounding box. They are asked to write an English expression to refer to the goal object so that it can be easily distinguished. Two expressions are collected for each object, each time with different distractor objects.

To ensure the model can distinguish between objects of different categories, we randomly select half the data, and replace two distractor objects in the group with two objects from a different category. Since these objects are from a different category, we expect the referring expression to still be able to correctly identify the goal object.

Each instance, or dataset sample, now consists of four objects (represented by images) with a query expression referring to the goal object. To associate each object with a *past expression*, we use expressions that were used to reference that object in other instances. Each instance now has a set of objects associated with a past expression and an image. In addition, the goal object is labeled and a query referring expression is provided for the goal. We randomly split the data into train, development and test sets in a 60/20/20 ratio. We refer to this split as STANDARD.

To evaluate the ability of grounding models to disambiguate objects from categories not seen during training, we create an alternative split of the *Diagnostic* dataset. This split ensures that no object category in the test set is present in the training or development splits. We refer to this split as HARD. More details of dataset construction and verification are present in the supplementary material.

3.2 Episodic Dataset

The *Diagnostic* dataset uses cropped images of objects from MSCOCO images and programmatically assigned past expressions. In order to capture nuances of realistic interaction, we collect a smaller *Episodic* dataset where annotators are shown a full scene and repeatedly asked to refer to objects within that scene. We select scenes from the MSCOCO dataset which have three to six objects of the same category. We prune ob-

[3] https://www.figure-eight.com/

ject bounding boxes with area less than 5% or over 50% of the image area. Extremely small objects are often not distinguishable, and large bounding boxes often correspond to a cluster of objects rather than a single object in cluttered scenes.

Each annotator is shown the same scene 10 times in a row. Each time one of the ambiguous objects is marked with a red bounding box. The annotator is asked to provide an English referring expression to uniquely identify the marked object. Our interface does not allow the user to view referring expressions once it has been entered. As a result, if the same object is marked again in the series, the user will have no way to look up what they had written before. This process simulates how a user will refer to objects in the same environment over a period of time. As the annotators do not communicate with each other, the dataset will not capture between-user entrainment. For each image, we collect two such series of 10 expressions from two different users. We create examples for the *Episodic* dataset in two ways:

SAME USER: For each scene and annotator, we order the expressions in the same order they were provided by the annotator. Each expression forms an example where it is the respective query expression and the candidate objects are all the objects in the scene. All previous referring expressions for this scene and annotator are assigned to the respective objects as past referring expressions. These examples capture the scenario where the interaction history is provided by a single user.

ACROSS USERS: Similar to the SAME USER dataset, we create a new example each time the annotator refers to an object. However, the past expressions come from the other annotator who was displayed the same scene. These examples represent cases where interaction history is acquired from people who do not interact with each other.

We create train and development sets with both types of examples. For testing, we create one set corresponding to each of the previously mentioned types. Validation details are in the supplementary material.

The statistics for both datasets are given in Table 1. We report a metric called *Lexical Overlap* to denote the extent of similarity between training and test data. The *Lexical Overlap* is the fraction of word types in the test set that also appear in the training set. As seen in Table 1, the HARD split has lower Lexical Overlap compared to STANDARD.

Dataset	Diagnostic		Episodic	
	STANDARD	HARD	SAME USER	ACROSS USERS
# Train	10133	9855	2656	2656
# Dev	3889	4170	714	714
# Test	3796	3793	1686	1686
Objects per Example	4.0	4.0	5.64	5.64
Expressions per Object	0.47	0.47	0.5	1.94
Lexical Overlap	0.71	0.63	0.47	0.47

Table 1: Statistics for the various datasets used. Unless otherwise mentioned, the numbers are reported from the test set. Low lexical overlap for HARD indicates more unseen words in the test set. The ACROSS USERS test set of Episodic dataset has more past expressions for each object compared to other splits.

This means that a greater number of novel words appear in the HARD dataset.

4 Experiments

We run multiple experiments to evaluate the proposed grounding models and characterize the advantages of using coreference along with visual features for grounding. Specifically, we consider the following questions:

1. Which method performs best for coreference grounding and the joint model? (Section 4.3)
2. How does grounding with different types of information (visual, coreference) transfer to new object categories? (Section 4.4)
3. How does grounding performance vary when past expressions are acquired from the same (or entrained) users, as opposed to users unknown to each other? (Section 4.5)

All quantitative evaluation can be found in Table 2. As our datasets contain examples where the object being referred to may or may not have a previous referring expression, we report overall test set accuracy (*All*), as well as accuracy grouped by the number of past referring expressions belonging to the ground truth object (*0, 1, or, 2*). This allows us to more accurately evaluate the coreference models as they are not expected to perform well without a previous referring expression.

We show how coreference grounding is used in practice by demonstrating its use on the Baxter robot. Namely, we show how a system can keep track of past referring expressions and associate them with objects.

4.1 Model Descriptions

We compare against the following unsupervised coreference baselines (i.e., not trained on the referring expression task). All these methods use a

similarity score between the input expression and the past referring expression to make a prediction.

Word Overlap Baseline: Compute the *Jaccard* similarity between two expressions.
Word Averaging Baseline: Each expression is represented by the average of the word vectors of constituent words. Similarity is computed as cosine similarity between vectors.
Paragram Phrase: Uses the paraphrase model proposed by Wieting et al. (2016) to compute similarity between past and input expressions.
InferSent Unsupervised: Each expression is represented by its pretrained *InferSent* embedding and similarity is computed as cosine similarity.

We also consider supervised models for the referring expression grounding task:

Vision: The *Compositional Modular Network* (Hu et al., 2017) which grounds an input expression to a bounding box's visual features.
Coreference: The proposed model that grounds an expression to objects represented by previous referring expressions. We evaluate the LSTM, BiLSTM-Max, Attention, and InferSent Encoders.
Joint: Jointly trains the *Vision* and *Coreference* components of the model. This model uses only the InferSent encoder. We evaluate both addition and concatenation methods for information fusion.

4.2 Implementation Details

Since the *Episodic* dataset is much smaller in size, all models (except joint with concatenation) are first trained on the *Diagnostic* training set until the validation error stops decreasing, and then trained on the *Episodic* train set. For the joint model with concatenation, we found that tuning only the fusion parameters, while holding the others fixed, on the *Episodic* data performs better.

The large *Diagnostic* dataset contains at most one past expression for each object. We cannot expect our models trained on the *Diagnostic* data to handle multiple past referring expressions. As a result, when an object is associated with multiple past expressions, we only consider the expression most similar to the query expression according to the Unsupervised InferSent model.

The models are implemented in *PyTorch* (Paszke et al., 2017). All models for the *Diagnostic* dataset are trained with the *Adam* optimizer (Kingma and Ba, 2015), and the best model is selected based on the performance on the development set. We use GloVe vectors (Pennington et al., 2014) and a pretrained VGG19 model to extract visual features (Simonyan and Zisserman, 2015).

4.3 Coreference and Joint Model Evaluation

We evaluate the performance of the coreference and joint models on the STANDARD split of the *Diagnostic* dataset (see column STANDARD of Table 2). First, we observe that all unsupervised coreference methods perform poorly when the goal object does not have past referring expressions. However, when past expressions are present, all the coreference baselines outperform the vision model. Coreference using pretrained InferSent embeddings performs the best among the unsupervised methods, possibly because InferSent embeddings have been pretrained on the SNLI dataset. SNLI was created from image captions, making the domain similar to that of referring expressions.

Learned models of coreference start performing better on cases where the goal object has no past referring expression. Note that in the 0-column, the 0 only refers to the ground-truth object not having a referring expression – the other objects may have expressions associated with them. Thus the supervised models can better determine when two expressions are incompatible, leading to the model choosing an object without any previous referring expression. However, when a past referring expression is present, they are typically informative and the *Word Overlap* model performs the best amongst unsupervised and supervised methods (see the 1-column). *InferSent (Unsupervised)* still achieves strong performance yet fine-tuning to the task still helps. We use the coreference model with the *InferSent* encoder for the joint models.

The jointly trained models achieve high accuracies in both cases where the ground truth object has previously been referred to (1-column) and those where it has not (0-column). This indicates that the models can successfully utilize information from both vision and coreference modalities. The concatenation fusion method outperform simple shallow addition of component scores.[4]

The joint model consistently outperforms the vision model. This is because people usually provide information relevant to how the object can be referred. If this information is available, which is

[4]Note that different objects in a scene might have different number of past expressions. At test time, we will not know the goal object, and hence, we cannot use the number of past expressions to determine which model to use at test time.

		Diagnostic						Episodic (*s. 4.5*)				
		STANDARD (*s. 4.3*)			HARD (*s. 4.4*)			SAME USER				ACROSS USERS
		0 57%	1 43%	All	0 56%	1 44%	All	0 49%	1 33%	2 15%	All	
	Vision	64.2	66.1	65.0	59.6	60.6	60.0	32.1	32.1	29.4	31.5	31.5
Unsupervised	Word Overlap	3.5	74.2	34.1	4.0	72.6	34.3	5.3	85.4	87.3	37.6	58.1
	Word Averaging	3.5	69.6	32.1	4.0	67.8	32.2	5.3	77.8	80.2	42.7	51.8
	Paragram Phrase	3.5	71.3	32.8	4.0	71.3	33.7	5.3	81.1	83.7	44.3	56.9
	InferSent	8.8	73.8	36.9	9.2	73.6	37.6	5.9	85.0	85.7	46.3	47.7
Supervised	LSTM	28.2	41.1	33.8	28.2	40.3	33.5	6.0	69.4	63.9	37.2	46.8
	BiLSTM-Max	30.0	60.8	43.3	27.0	57.8	40.6	7.0	75.7	74.6	41.8	53.1
	Attention	29.2	61.6	43.2	26.8	56.0	39.7	10.1	67.0	63.9	38.4	47.4
	InferSent	27.9	70.5	46.3	28.8	68.6	46.4	5.8	82.5	85.3	45.4	61.3
Joint	Addition	65.3	69.4	67.1	62.0	62.3	62.1	22.3	63.8	60.3	42.6	48.7
	Concatenation	68.6	71.3	**69.8**	58.0	70.7	**63.6**	19.0	84.7	86.1	**52.7**	**62.7**

Table 2: Grounding accuracy under different conditions. Best scores in bold. The columns labeled **0**, **1**, and **2** correspond to test examples where the goal object has 0, 1 and 2 past expressions respectively (percentage indicates fraction of test examples applicable for that column). **All** refers to the score on the entire test set. Most examples in ACROSS USERS have large number of past expressions, so we only report score on the entire test set.

true in many realistic scenarios, it is beneficial to utilize coreference grounding.

With more data, the vision model could achieve higher performance. However, we argue that as ambiguity between objects increases, the properties that distinguish objects become more subtle and a large dataset would be necessary to learn these intricacies.

When does Joint perform better than Vision and Coreference? Our *Joint* model performs better than models trained on single modality (*Vision*, *Coreference*). The joint model can use visual features when the past referring expressions are not sufficient to discriminate between objects. In 25% of examples, the *Coreference* model predicts the wrong object whereas the *Joint* model selects the correct object. A majority of these cases are examples where the goal object has no past expression associated with it. In 8% of examples, the *Joint* and *Coreference* models are correct even though the *Vision* model is wrong. Finally, for only 7% of the examples, the *Joint* model predicts the correct object when both the *Vision* and *Coreference* models predict incorrectly. This indicates that the *Joint* model is primarily merging the gains of the *Vision* and *Coreference* models; most correct decisions of *Joint* correspond to correct decisions either from *Vision* or *Coreference*. Figure 3 shows a breakdown of how often the various models outperform each other. Figure 4 shows examples where Joint outperforms models trained on a single modality.

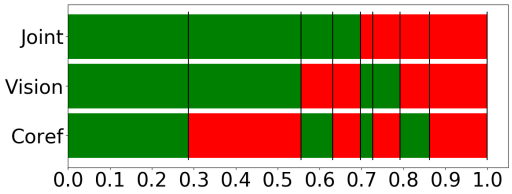


Figure 3: Proportion of examples of the STANDARD test set where different subsets of the systems ground correctly (green) or incorrectly (red). Instances are arranged along the x-axis.

4.4 Generalizing to New Object Categories

To evaluate how the various models handle generalization to unseen object categories, we use the HARD split of the *Diagnostic* dataset. The test set of this split contains object categories which have not explicitly been seen during training. We hypothesize that due to pretrained word embeddings (which include embeddings for words describing the unknown categories), the coreference models will be able to successfully ground to new object categories. On the other hand, the performance of the *Vision* model will decrease in the HARD split, because the pretrained visual features of the new objects are not well aligned with representations of unseen words. As seen in Table 2, this is indeed the case as the *Vision* model's performance drops between the STANDARD and HARD datasets. On the other hand, the coreference models' performance on the HARD split is comparable to those

of the STANDARD split. Although the aggregate performance is low, the coreference models perform strongly on examples where the goal object has past expressions (see 1-column). In particular, *Coref Supervised with InferSent* achieves 70.5% on the STANDARD split, which reduces only to 68.6% on the HARD split. This can be explained by observing that the representation of the object (its past referring expression) and a new referring expression are already aligned within the same vector space (pretrained InferSent embeddings).

4.5 Performance on Episodic Dataset

As the *Diagnostic* dataset is somewhat artificial in the way objects and properties are grouped, we also evaluate these models on the *Episodic* dataset. As described in Section 3.2, the key differences in this dataset are that all candidate objects in an example are from the *same* scene, and previous referring expressions are added sequentially as the user refers to more objects in the scene (thus a previous referring expression truly did occur *previously*).

We find similar trends in the performance on the *Episodic* dataset (see the *Episodic* columns of Table 2). Since the same user is referring to the same object multiple times, we expect expressions for the same object to be similar. This leads to the high performance of coreference models in the SAME USER split of the *Episodic* dataset (see the 1-column). Even the word-overlap model does particularly well due to the correlation between expressions from the same user (over 85% accuracy when the goal object has past expressions).

If the past expressions were not provided by the same user, but by users unknown to each other, we can still see improvement over not having any past expressions (see the ACROSS USERS column). In this split, past expressions are always associated with objects as we assume other users have interacted with the system. In the ACROSS USERS split, even though the *Coreference* models are less effective than in the SAME USER split, they still outperform the *Vision* model (e.g., the *Supervised InferSent* model achieves 61.3%, whereas a pure vision model achieves only 31.5%). This is because users unknown to each other still provide useful knowledge about the properties of objects.

The *Joint* models benefit from using both modalities, which is particularly important in realistic scenarios as most objects initially will not have any previous referring expressions. Note

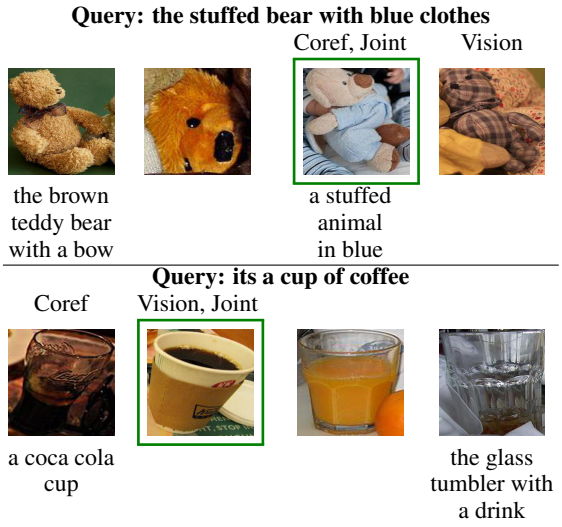


Figure 4: Examples where the Joint model accurately grounds the query, but either Vision or Coreference does not. The past expression is mentioned below each object image. The correct object has a green border and a model's name appears above its predicted object.

that the performance of the *Vision* model on the *Episodic* dataset is much lower than its performance on the *Diagnostic* dataset. This loss in performance is due to the images in the *Episodic* dataset being more cluttered and annotators often using spatial relationships to refer. As the *Diagnostic* training dataset does not contain spatial information, our models cannot handle these cases. We can train on existing referring expression datasets to learn spatial relationships, but we do not explore this as it is not central to this paper. More analysis is added to the appendix.

4.6 Robot Demonstration

One of the motivations of this work was to enable robots to use past referring expressions to aid grounding in human-robot interactions. In this section, we provide a demonstration of how our coreference grounding system can be integrated with the Baxter robotic platform. To benefit from coreference grounding, the system must have a natural way to keep track of past referring expressions of an object. We demonstrate such an interface in Figure 5.

We focus on a scenario where a human asks the robot to pick up specific objects. If the robot incorrectly identifies the object being referred to, on the next turn, the user can correct the robot by indicating the correct object (Figure 5b). The referring expression can then be associated with the correct

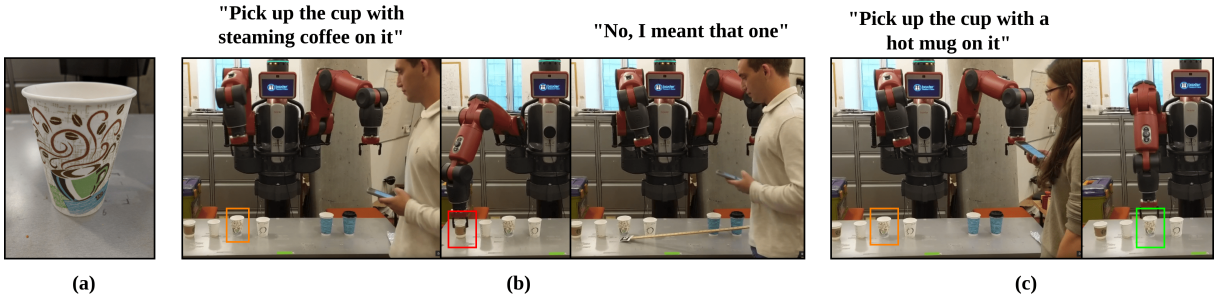


Figure 5: Demonstration of a coreference grounding system on the Baxter robot. (a) The cup being identified. (b) A user refers to the cup and the robot grounds incorrectly. The user corrects the robot and associates the referring expression with that object. (c) When a new user refers to the same object, the system's output is correct.

object and this association can be maintained with a tracking system. Future users (with no knowledge of the first user) can then successfully refer to the object (Figure 5c).

5 Related Work

Grounding Referring Expressions There has been a lot of recent work on resolving referring expressions to objects (Mao et al., 2016; Yu et al., 2016; Shridhar and Hsu, 2018; Nagaraja et al., 2016; Cirik et al., 2018). In contrast to these approaches which are static once trained, our model leverages past referring expressions, which permits an interface to add information during execution. Our task is related to *Visual Dialogue* (Das et al., 2017; Kottur et al., 2018), where an agent interactively answers questions about visual properties of a scene where the answers only exist in the image and not in the dialogue context. In contrast, we focus on using complementary knowledge from past referring expressions. There has also been work to learn a compatibility metric dictating whether two words can refer to the same object (Kruszewski and Baroni, 2015). Our work focuses on coreference between entire referring expressions instead of atomic words.

Lexical Choice in Interactions A large body of work in psycholinguistics (Clark and Wilkes-Gibbs, 1986; Brennan and Clark, 1996; Pickering and Garrod, 2004) show that participants in a conversation collaboratively come up with lexical terms to refer to objects and consequently, get *entrained* with each other and start using similar expressions to refer to objects. These works form the motivation for our problem formulation. Recently, the *PhotoBook Dataset* has been proposed to investigate shared dialogue history in conversation (Haber et al., 2019). The dataset differs from ours in that expressions refer to entire scenes instead of objects within a scene. The authors' conclusions support our findings that using past expressions is useful for resolving referring expressions.

Interaction History for Human Robot Interaction Paul et al. (2017) maintains knowledge provided by users, using a closed set of predicates. In constrast, we use raw past referring expressions to handle an open domain of knowledge. There has been work on grounding in dialogue for robots (Tellex et al., 2014; Whitney et al., 2017; Thomason et al., 2017; Padmakumar et al., 2017). In contrast to our work, they focus on refinement or clarification, and hence past expressions only help within the same dialogue episode.

6 Conclusion

In this work, we reformulated the grounding problem as a type of coreference resolution, allowing for the inclusion of past referring expressions that are typical in many real-world scenarios. We proposed a model that can use both linguistic features from past expressions and visual features of the object to ground to a new expression. We showed that this model outperforms a purely vision-based model as it can use past descriptions of salient features that the vision-based model may have difficulty learning with limited data. It further benefits from having a vision model that can fill in information not provided in past expressions.

Acknowledgements

We gratefully acknowledge funding support in part by the Honda Research Institute and the Toyota Research Institute. However, this article solely reflects the opinions and conclusions of its authors.

References

Susan E. Brennan and Herbert H. Clark. 1996. Conceptual pacts and lexical choice in conversation. In *Journal of Experimental Psychology: Learning, Memory, and Cognition*.

Volkan Cirik, Taylor Berg-Kirkpatrick, and Louis-Philippe Morency. 2018. Using syntax to ground referring expressions in natural images. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

Herbert H. Clark and Deanna Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22(1):1–39.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M.F. Moura, Devi Parikh, and Dhruv Batra. 2017. Visual Dialog. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.

Janosch Haber, Tim Baumgärtner, Ece Takmaz, Lieke Gelderloos, Elia Bruni, and Raquel Fernández. 2019. The photobook dataset: Building common ground through visually-grounded dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. 2017. Modeling relationships in referential expressions with compositional modular networks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. In *Proceedings of the 15th European Conference on Computer Vision (ECCV)*.

Germán Kruszewski and Marco Baroni. 2015. So similar and yet incompatible: Toward the automated identification of semantically compatible words. In *HLT-NAACL*.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Proceedings of the 13th European Conference on Computer Vision (ECCV)*.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.

Varun K. Nagaraja, Vlad I. Morariu, and Larry S. Davis. 2016. Modeling context between objects for referring expression understanding. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*.

Aishwarya Padmakumar, Jesse Thomason, and Raymond J. Mooney. 2017. Integrated learning of dialog strategies and semantic parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Rohan Paul, Andrei Barbu, Sue Felshin, Boris Katz, and Nicholas Roy. 2017. Temporal grounding graphs for language understanding with accrued visual-linguistic context. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Martin J. Pickering and Simon Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences*, 27(2):169–190.

Mohit Shridhar and David Hsu. 2018. Interactive visual grounding of referring expressions for human-robot interaction. In *Proceedings of Robotics: Science and Systems (RSS)*.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Stefanie Tellex, Ross Knepper, Adrian Li, Daniela Rus, and Nicholas Roy. 2014. Asking for Help Using Inverse Semantics. In *Proceedings of Robotics: Science and Systems (RSS)*.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Justin Hart, Peter Stone, and Raymond J. Mooney. 2017. Opportunistic active learning for grounding natural language descriptions. In *Proceedings of the 1st Conference on Robot Learning (CoRL)*.

David Whitney, Eric Rosen, James MacGlashan, Lawson Wong, and Stefanie Tellex. 2017. Reducing Errors in Object-Fetching Interactions through Social Feedback. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*.

Licheng Yu, Patric Poirson, Shan Yang, Alexander C. Berg, and Tamara L. Berg. 2016. Modeling context in referring expressions. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*.

Procedural Reasoning Networks for Understanding Multimodal Procedures

Mustafa Sercan Amac **Semih Yagcioglu** **Aykut Erdem** **Erkut Erdem**
Hacettepe University Computer Vision Lab
Dept. of Computer Engineering, Hacettepe University, Ankara, TURKEY
{b21626915,n13242994,aykut,erkut}@cs.hacettepe.edu.tr

Abstract

This paper addresses the problem of comprehending procedural commonsense knowledge. This is a challenging task as it requires identifying key entities, keeping track of their state changes, and understanding temporal and causal relations. Contrary to most of the previous work, in this study, we do not rely on strong inductive bias and explore the question of how multimodality can be exploited to provide a complementary semantic signal. Towards this end, we introduce a new entity-aware neural comprehension model augmented with external relational memory units. Our model learns to dynamically update entity states in relation to each other while reading the text instructions. Our experimental analysis on the visual reasoning tasks in the recently proposed RecipeQA dataset reveals that our approach improves the accuracy of the previously reported models by a large margin. Moreover, we find that our model learns effective dynamic representations of entities even though we do not use any supervision at the level of entity states.[1]

1 Introduction

A great deal of commonsense knowledge about the world we live is procedural in nature and involves steps that show ways to achieve specific goals. Understanding and reasoning about procedural texts (*e.g.* cooking recipes, how-to guides, scientific processes) are very hard for machines as it demands modeling the intrinsic dynamics of the procedures (Bosselut et al., 2018; Dalvi et al., 2018; Yagcioglu et al., 2018). That is, one must be aware of the entities present in the text, infer relations among them and even anticipate changes in the states of the entities after each action. For example, consider the cheeseburger recipe presented in Fig. 1. The instruction "*salt and pepper each patty and cook for 2 to 3 minutes on the first side*" in Step 5 entails mixing three basic ingredients, the *ground beef*, *salt* and *pepper*, together and then applying heat to the mix, which in turn causes chemical changes that alter both the appearance and the taste. From a natural language understanding perspective, the main difficulty arises when a model sees the word *patty* again at a later stage of the recipe. It still corresponds to the same entity, but its form is totally different.

Over the past few years, many new datasets and approaches have been proposed that address this inherently hard problem (Bosselut et al., 2018; Dalvi et al., 2018; Tandon et al., 2018; Du et al., 2019). To mitigate the aforementioned challenges, the existing works rely mostly on heavy supervision and focus on predicting the individual state changes of entities at each step. Although these models can accurately learn to make local predictions, they may lack global consistency (Tandon et al., 2018; Du et al., 2019), not to mention that building such annotated corpora is very labor-intensive. In this work, we take a different direction and explore the problem from a multimodal standpoint. Our basic motivation, as illustrated in Fig. 1, is that accompanying images provide complementary cues about causal effects and state changes. For instance, it is quite easy to distinguish raw meat from cooked one in visual domain.

In particular, we take advantage of recently proposed RecipeQA dataset (Yagcioglu et al., 2018), a dataset for multimodal comprehension of cooking recipes, and ask whether it is possible to have a model which employs dynamic representations of entities in answering questions that require multimodal understanding of procedures. To this end, inspired from (Santoro et al., 2018), we propose Procedural Reasoning Networks (PRN) that incorporates entities into the comprehension process and al-

[1]The project website with code and demo is available at https://hucvl.github.io/prn/

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 441–451
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics



Figure 1: A recipe for preparing a cheeseburger (adapted from the cooking instructions available at https://www.instructables.com/id/In-N-Out-Double-Double-Cheeseburger-Copycat). Each basic ingredient (entity) is highlighted by a different color in the text and with bounding boxes on the accompanying images. Over the course of the recipe instructions, ingredients interact with each other, change their states by each cooking action (underlined in the text), which in turn alter the visual and physical properties of entities. For instance, the *tomato* changes it form by being *sliced up* and then *stacked* on a *hamburger bun*.

lows to keep track of entities, understand their interactions and accordingly update their states across time. We report that our proposed approach significantly improves upon previously published results on visual reasoning tasks in RecipeQA, which test understanding causal and temporal relations from images and text. We further show that the dynamic entity representations can capture semantics of the state information in the corresponding steps.

2 Visual Reasoning in RecipeQA

In our study, we particularly focus on the visual reasoning tasks of RecipeQA, namely *visual cloze*, *visual coherence*, and *visual ordering* tasks, each of which examines a different reasoning skill[2]. We briefly describe these tasks below.

Visual Cloze. In the visual cloze task, the question is formed by a sequence of four images from consecutive steps of a recipe where one of them is replaced by a placeholder. A model should select the correct one from a multiple-choice list of four answer candidates to fill in the missing piece. In that regard, the task inherently requires aligning visual and textual information and understanding temporal relationships between the cooking actions and the entities.

Visual Coherence. The visual coherence task tests the ability to identify the image within a sequence of four images that is inconsistent with the text instructions of a cooking recipe. To succeed in this task, a model should have a clear understanding of the procedure described in the recipe and at the same time connect language and vision.

Visual Ordering. The visual ordering task is about grasping the temporal flow of visual events with the help of the given recipe text. The questions show a set of four images from the recipe and the task is to sort jumbled images into the correct order. Here, a model needs to infer the temporal relations between the images and align them with the recipe steps.

3 Procedural Reasoning Networks

In the following, we explain our Procedural Reasoning Networks model. Its architecture is based on a bi-directional attention flow (BiDAF) model (Gardner et al., 2018)[3], but also equipped with an explicit reasoning module that acts on entity-specific rela-

[2]We intentionally leave the textual cloze task out from our experiments as the questions in this task does not necessarily need multimodality.

[3]Our implementation is based on the implementation publicly available in AllenNLP (Gardner et al., 2018).

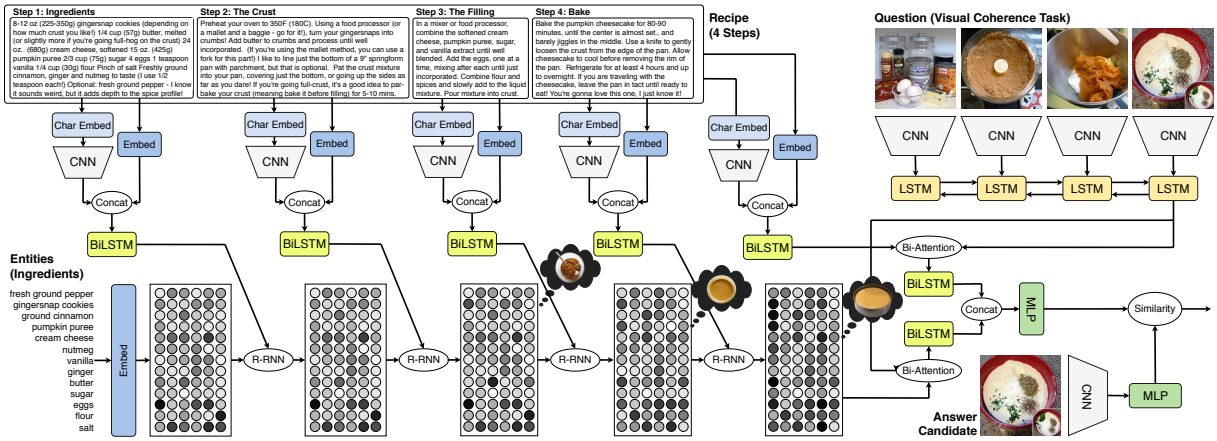


Figure 2: An illustration of our Procedural Reasoning Networks (PRN). For a sample question from visual coherence task in RecipeQA, while reading the cooking recipe, the model constantly performs updates on the representations of the entities (ingredients) after each step and makes use of their representations along with the whole recipe when it scores a candidate answer. Please refer to the main text for more details.

tional memory units. Fig. 2 shows an overview of the network architecture. It consists of five main modules: An input module, an attention module, a reasoning module, a modeling module, and an output module. Note that the question answering tasks we consider here are multimodal in that while the context is a procedural text, the question and the multiple choice answers are composed of images.

1. **Input Module** extracts vector representations of inputs at different levels of granularity by using several different encoders.
2. **Reasoning Module** scans the procedural text and tracks the states of the entities and their relations through a recurrent relational memory core unit (Santoro et al., 2018).
3. **Attention Module** computes context-aware query vectors and query-aware context vectors as well as query-aware memory vectors.
4. **Modeling Module** employs two multi-layered RNNs to encode previous layers outputs.
5. **Output Module** scores a candidate answer from the given multiple-choice list.

At a high level, as the model is reading the cooking recipe, it continually updates the internal memory representations of the entities (ingredients) based on the content of each step – it keeps track of changes in the states of the entities, providing an entity-centric summary of the recipe. The response to a question and a possible answer depends on the representation of the recipe text as well as the last states of the entities. All this happens in a series of implicit relational reasoning steps and there is no need for explicitly encoding the state in terms of a predefined vocabulary.

3.1 Input Module

Let the triple $(\mathbf{R}, \mathbf{Q}, \mathbf{A})$ be a sample input. Here, $\mathbf{R}$ denotes the input recipe which contains textual instructions composed of N words in total. $\mathbf{Q}$ represents the question that consists of a sequence of M images. $\mathbf{A}$ denotes an answer that is either a single image or a series of L images depending on the reasoning task. In particular, for the visual cloze and the visual coherence type questions, the answer contains a single image ($L = 1$) and for the visual ordering task, it includes a sequence.

We encode the input recipe $\mathbf{R}$ at character, word, and step levels. Character-level embedding layer uses a convolutional neural network, namely Char-CNN model by Kim (2014), which outputs character level embeddings for each word and alleviates the issue of out-of-vocabulary (OOV) words. In word embedding layer, we use a pretrained GloVe model (Pennington et al., 2014) and extract word-level embeddings[4]. The concatenation of the character and the word embeddings are then fed to a two-layer highway network (Srivastava et al., 2015) to obtain a contextual embedding for each word in the recipe. This results in the matrix $\mathbf{R}' \in \mathbb{R}^{2d \times N}$.

On top of these layers, we have another layer that encodes the steps of the recipe in an individual manner. Specifically, we obtain a step-level con-

[4]We also consider pretrained ELMo embeddings (Peters et al., 2018) in our experiments but found out that the performance gain does not justify the computational overhead.

textual embedding of the input recipe containing T steps as $\mathcal{S} = (\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_T)$ where $\mathbf{s}_i$ represents the final state of a BiLSTM encoding the i-th step of the recipe obtained from the character and word-level embeddings of the tokens exist in the corresponding step.

We represent both the question $\mathbf{Q}$ and the answer $\mathbf{A}$ in terms of visual embeddings. Here, we employ a pretrained ResNet-50 model (He et al., 2016) trained on ImageNet dataset (Deng et al., 2009) and represent each image as a real-valued 2048-d vector using features from the penultimate average-pool layer. Then these embeddings are passed first to a multilayer perceptron (MLP) and then its outputs are fed to a BiLSTM. We then form a matrix $\mathbf{Q}' \in \mathbb{R}^{2d \times M}$ for the question by concatenating the cell states of the BiLSTM. For the visual ordering task, to represent the sequence of images in the answer with a single vector, we additionally use a BiLSTM and define the answering embedding by the summation of the cell states of the BiLSTM. Finally, for all tasks, these computations produce answer embeddings denoted by $\mathbf{a} \in \mathbb{R}^{2d \times 1}$.

3.2 Reasoning Module

As mentioned before, comprehending a cooking recipe is mostly about entities (basic ingredients) and actions (cooking activities) described in the recipe instructions. Each action leads to changes in the states of the entities, which usually affects their visual characteristics. A change rarely occurs in isolation; in most cases, the action affects multiple entities at once. Hence, in our reasoning module, we have an explicit memory component implemented with relational memory units (Santoro et al., 2018). This helps us to keep track of the entities, their state changes and their relations in relation to each other over the course of the recipe (see Fig. 3). As we will examine in more detail in Section 4, it also greatly improves the interpretability of model outputs.

Specifically, we set up the memory with a memory matrix $\mathbf{E} \in \mathbb{R}^{d_E \times K}$ by extracting K entities (ingredients) from the first step of the recipe[5]. We initialize each memory cell $\mathbf{e}_i$ representing a specific entity by its CharCNN and pre-trained GloVe embeddings[6]. From now on, we will use the terms memory cells and entities interchangeably throughout the paper. Since the input recipe is given in the form of a procedural text decomposed into a number of steps, we update the memory cells after each step, reflecting the state changes happened on the entities. This update procedure is modelled via a relational recurrent neural network (R-RNN), recently proposed by Santoro et al. (2018). It is built on a 2-dimensional LSTM model whose matrix of cell states represent our memory matrix $\mathbf{E}$. Here, each row i of the matrix $\mathbf{E}$ refers to a specific entity $\mathbf{e}_i$ and is updated after each recipe step t as follows:

$$\phi_{i,t} = \text{R-RNN}(\phi_{i,t-1}, \mathbf{s}_t) \tag{1}$$

where $\mathbf{s}_t$ denotes the embedding of recipe step t and $\phi_{i,t} = (\mathbf{h}_{i,t}, \mathbf{e}_{i,t})$ is the cell state of the R-RNN at step t with $\mathbf{h}_{i,t}$ and $\mathbf{e}_{i,t}$ being the i-th row of the hidden state of the R-RNN and the dynamic representation of entity $\mathbf{e}_i$ at the step t, respectively. The R-RNN model exploits a multi-headed self-attention mechanism (Vaswani et al., 2017) that allows memory cells to interact with each other and attend multiple locations simultaneously during the update phase.

In Fig. 3, we illustrate how this interaction takes place in our relational memory module by considering a sample cooking recipe and by presenting how the attention matrix changes throughout the recipe. In particular, the attention matrix at a specific time shows the attention flow from one entity (memory cell) to another along with the attention weights to the corresponding recipe step (offset column). The color intensity shows the magnitude of the attention weights. As can be seen from the figure, the internal representations of the entities are actively updated at each step. Moreover, as argued in (Santoro et al., 2018), this can be interpreted as a form of relational reasoning as each update on a specific memory cell is operated in relation to others. Here, we should note that it is often difficult to make sense of these attention weights. However, we observe that the attention matrix changes very gradually near the completion of the recipe.

3.3 Attention Module

Attention module is in charge of linking the question with the recipe text and the entities present in the recipe. It takes the matrices $\mathbf{Q}'$ and $\mathbf{R}'$ from the input module, and $\mathbf{E}$ from the reasoning module

[5]The first steps of the recipes in RecipeQA commonly contain a list of ingredients.

[6]Multi-word entities (*e.g. minced garlic*) are represented by the average embedding vector of the words that they contain, and OOV words are expressed with the average word vector of all the words.

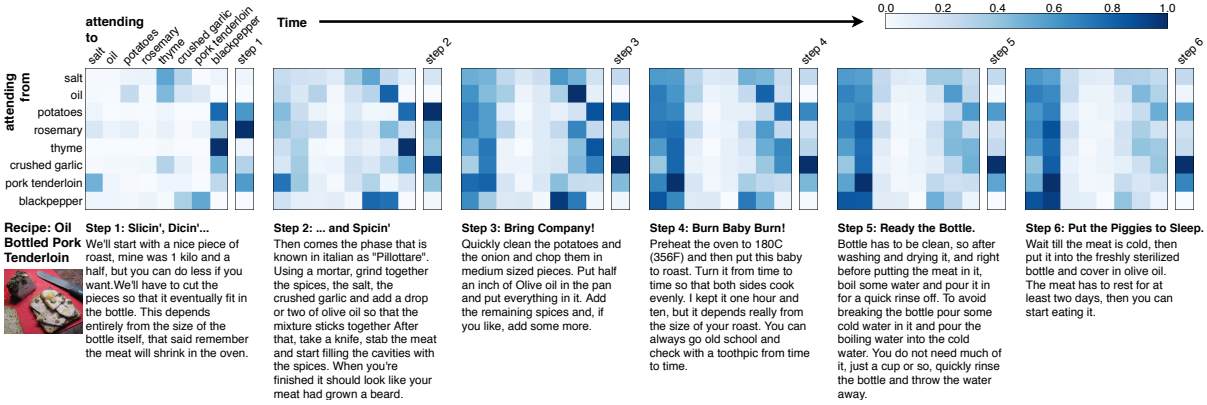


Figure 3: Sample visualizations of the self-attention weights demonstrating both the interactions among the ingredients and between the ingredients and the textual instructions throughout the steps of a sample cooking recipe from RecipeQA (darker colors imply higher attention weights). The attention maps do not change much after the third step as the steps after that mostly provide some redundant information about the completed recipe.

and constructs the question-aware recipe representation $\mathbf{G}$ and the question-aware entity representation $\mathbf{Y}$. Following the attention flow mechanism described in (Seo et al., 2017a), we specifically calculate attentions in four different directions: (1) from question to recipe, (2) from recipe to question, (3) from question to entities, and (4) from entities to question.

The first two of these attentions require computing a shared affinity matrix $\mathbf{S}^R \in \mathbb{R}^{N \times M}$ with $\mathbf{S}^R_{i,j}$ indicating the similarity between i-th recipe word and j-th image in the question estimated by

$$\mathbf{S}^R_{i,j} = \mathbf{w}_R^\top [\mathbf{R}'_i; \mathbf{Q}'_j; \mathbf{R}'_i \circ \mathbf{Q}'_j] \tag{2}$$

where $\mathbf{w}_R^\top$ is a trainable weight vector, $\circ$ and $[;]$ denote elementwise multiplication and concatenation operations, respectively.

Recipe-to-question attention determines the images within the question that is most relevant to each word of the recipe. Let $\tilde{\mathbf{Q}} \in \mathbb{R}^{2d \times N}$ represent the recipe-to-question attention matrix with its i-th column being given by $\tilde{\mathbf{Q}}_i = \sum_j \mathbf{a}_{ij} \mathbf{Q}'_j$ where the attention weight is computed by $\mathbf{a}_i = \mathrm{softmax}(\mathbf{S}^R_i) \in \mathbb{R}^M$.

Question-to-recipe attention signifies the words within the recipe that have the closest similarity to each image in the question, and construct an attended recipe vector given by $\tilde{\mathbf{r}} = \sum_i \mathbf{b}_i \mathbf{R}'_i$ with the attention weight is calculated by $\mathbf{b} = \mathrm{softmax}(\mathrm{max}_{col}(\mathbf{S}^R)) \in \mathbb{R}^N$ where max_{col} denotes the maximum function across the column. The question-to-recipe matrix is then obtained by replicating $\tilde{\mathbf{r}}$ N times across the column, giving $\tilde{\mathbf{R}} \in \mathbb{R}^{2d \times N}$.

Then, we construct the question aware representation of the input recipe, $\mathbf{G}$, with its i-th column $\mathbf{G}_i \in \mathbb{R}^{8d \times N}$ denoting the final embedding of i-th word given by

$$\mathbf{G}_i = [\mathbf{R}'_i; \tilde{\mathbf{Q}}_i; \mathbf{R}'_i \circ \tilde{\mathbf{Q}}_i; \mathbf{R}'_i \circ \tilde{\mathbf{R}}_i;] \ . \tag{3}$$

Attentions from question to entities, and from entities to question are computed in a way similar to the ones described above. The only difference is that it uses a different shared affinity matrix to be computed between the memory encoding entities $\mathbf{E}$ and the question $\mathbf{Q}'$. These attentions are then used to construct the question aware representation of entities, denoted by $\mathbf{Y}$, that links and integrates the images in the question and the entities in the input recipe.

3.4 Modeling Module

Modeling module takes the question-aware representations of the recipe $\mathbf{G}$ and the entities $\mathbf{Y}$, and forms their combined vector representation. For this purpose, we first use a two-layer BiLSTM to read the question-aware recipe $\mathbf{G}$ and to encode the interactions among the words conditioned on the question. For each direction of BiLSTM , we use its hidden state after reading the last token as its output. In the end, we obtain a vector embedding $\mathbf{c} \in \mathbb{R}^{2d \times 1}$. Similarly, we employ a second BiLSTM, this time, over the entities $\mathbf{Y}$, which results in another vector embedding $\mathbf{f} \in \mathbb{R}^{2d_E \times 1}$. Finally, these vector representations are concatenated and then projected to a fixed size representation using $\mathbf{o} = \varphi_o([\mathbf{c}; \mathbf{f}]) \in \mathbb{R}^{2d \times 1}$ where φ_o is a multilayer perceptron with $\tanh$ activation function.

3.5 Output Module

The output module takes the output of the modeling module, encoding vector embeddings of the question-aware recipe and the entities $\mathbf{Y}$, and the embedding of the answer $\mathbf{A}$, and returns a similarity score which is used while determining the correct answer. Among all the candidate answer, the one having the highest similarity score is chosen as the correct answer. To train our proposed procedural reasoning network, we employ a hinge ranking loss (Collobert et al., 2011), similar to the one used in (Yagcioglu et al., 2018), given below.

$$L = \max\{0, \gamma - \cos(\mathbf{o}, \mathbf{a}_+) + \cos(\mathbf{o}, \mathbf{a}_-)\} \quad (4)$$

where γ is the margin parameter, $\mathbf{a}_+$ and $\mathbf{a}_-$ are the correct and the incorrect answers, respectively.

4 Experiments

In this section, we describe our experimental setup and then analyze the results of the proposed Procedural Reasoning Networks (PRN) model.

4.1 Entity Extraction

Given a recipe, we automatically extract the entities from the initial step of a recipe by using a dictionary of ingredients. While determining the ingredients, we exploit Recipe1M (Marin et al., 2018) and Kaggle Whats Cooking Recipes (Yummly, 2015) datasets, and form our dictionary using the most commonly used ingredients in the training set of RecipeQA. For the cases when no entity can be extracted from the recipe automatically (20 recipes in total), we manually annotate those recipes with the related entities.

4.2 Training Details

In our experiments, we separately trained models on each task, as well as we investigated multi-task learning where a single model is trained to solve all these tasks at once. In total, the PRN architecture consists of ∼12M trainable parameters. We implemented our models in PyTorch (Paszke et al., 2017) using AllenNLP library (Gardner et al., 2018). We used Adam optimizer with a learning rate of 1e-4 with an early stopping criteria with the patience set to 10 indicating that the training procedure ends after 10 iterations if the performance would not improve. We considered a batch size of 32 due to our hardware constraints. In the multi-task setting, batches are sampled round-robin from all tasks, where each batch is solely composed of examples from one task. We performed our experiments on a system containing four NVIDIA GTX-1080Ti GPUs, and training a single model took around 2 hours. We employed the same hyperparameters for all the baseline systems. We plan to share our code and model implementation after the review process.

4.3 Baselines

We compare our model with several baseline models as described below. We note that the results of the first two are previously reported in (Yagcioglu et al., 2018).

Hasty Student (Yagcioglu et al., 2018) is a heuristics-based simple model which ignores the recipe and gives an answer by examining only the question and the answer set using distances in the visual feature space.

Impatient Reader (Hermann et al., 2015) is a simple neural model that takes its name from the fact that it repeatedly computes attention over the recipe after observing each image in the query.

BiDAF (Seo et al., 2017a) is a strong reading comprehension model that employs a bi-directional attention flow mechanism to obtain a question-aware representation and bases its predictions on this representation. Originally, it is a span-selection model from the input context. Here, we adapt it to work in a multimodal setting and answer multiple choice questions instead.

BiDAF w/ static memory is an extended version of the BiDAF model which resembles our proposed PRN model in that it includes a memory unit for the entities. However, it does not make any updates on the memory cells. That is, it uses the static entity embeeddings initialized with GloVe word vectors. We propose this baseline to test the significance of the use of relational memory updates.

4.4 Results

Table 1 presents the quantitative results for the visual reasoning tasks in RecipeQA. In single-task training setting, PRN gives state-of-the-art results compared to other neural models. Moreover, it achieves the best performance on average. These results demonstrate the importance of having a dynamic memory and keeping track of entities extracted from the recipe. In multi-task training set-

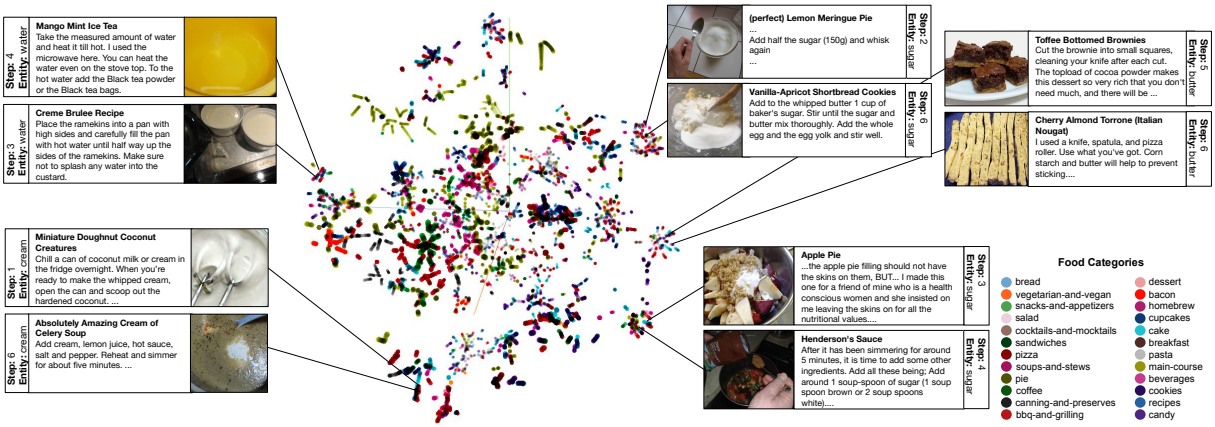


Figure 4: t-SNE visualizations of learned embeddings from each memory snapshot mapping to each entity and their corresponding states from each step for visual cloze task.

Model	Single-task Training				Multi-task Training			
	Cloze	Coherence	Ordering	Average	Cloze	Coherence	Ordering	All
Human*	77.60	81.60	64.00	74.40	–	–	–	–
Hasty Student	27.35	**65.80**	40.88	44.68	–	–	–	–
Impatient Reader	27.36	28.08	26.74	27.39	–	–	–	–
BIDAF	53.95	48.82	62.42	55.06	44.62	36.00	**63.93**	48.67
BIDAF w/ static memory	51.82	45.88	60.90	52.87	**47.81**	40.23	62.94	**50.59**
PRN	**56.31**	53.64	**62.77**	**57.57**	46.45	**40.58**	62.67	50.17

* Taken from the RecipeQA project website, based on 100 questions sampled randomly from the validation set.

Table 1: Quantitative comparison of the proposed PRN model against the baselines.

ting where a single model is trained to solve all the tasks at once, PRN and BIDAF w/ static memory perform comparably and give much better results than BIDAF. Note that the model performances in the multi-task training setting are worse than single-task performances. We believe that this is due to the nature of the tasks that some are more difficult than the others. We think that the performance could be improved by employing a carefully selected curriculum strategy (McCann et al., 2018).

In Fig. 4, we illustrate the entity embeddings space by projecting the learned embeddings from the step-by-step memory snapshots through time with t-SNE to 3-d space from 200-d vector space. Color codes denote the categories of the cooking recipes. As can be seen, these step-aware embeddings show clear clustering of these categories. Moreover, within each cluster, the entities are grouped together in terms of their state characteristics. For instance, in the zoomed parts of the figure, chopped and sliced, or stirred and whisked entities are placed close to each other.

Fig. 5 demonstrates the entity arithmetics using the learned embeddings from each entity step. Here, we show that the learned embedding from the memory snapshots can effectively capture the contextual information about the entities at each time point in the corresponding step while taking into account of the recipe data. This basic arithmetic operation suggests that the proposed model can successfully capture the semantics of each entity's state in the corresponding step[7].

5 Related Work

In recent years, tracking entities and their state changes have been explored in the literature from a variety of perspectives. In an early work, Henaff et al. (2017) proposed a dynamic memory based network which updates entity states using a gating mechanism while reading the text. Bansal et al. (2017) presented a more structured memory augmented model which employs memory slots for representing both entities and their relations. Pavez et al. (2018) suggested a conceptually similar model in which the pairwise relations between attended memories are utilized to encode the world

[7]We used Gensim for calculating entity arithmetics using cosine distances between entity embeddings.

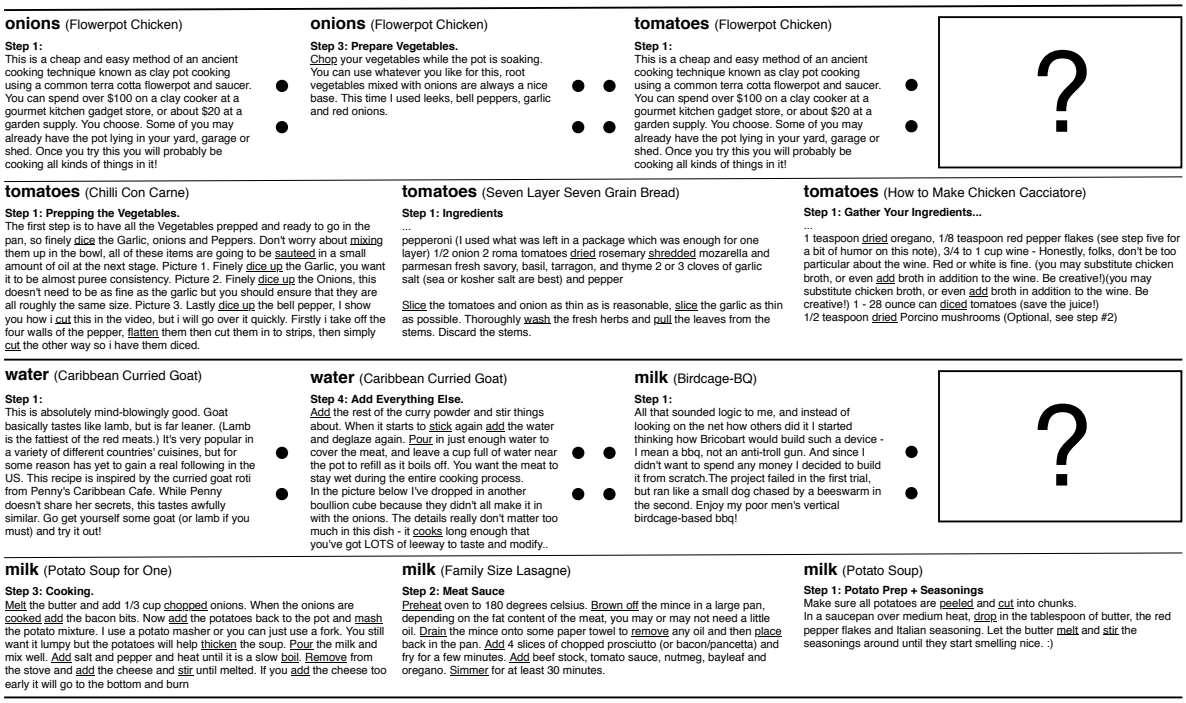


Figure 5: Step-aware entity representations can be used to discover the changes occurred in the states of the ingredients between two different recipe steps. The difference vector between two entities can then be added to other entities to find their next states. For instance, in the first example, the difference vector encodes the chopping action done on onions. In the second example, it encodes the pouring action done on the water. When these vectors are added to the representations of raw tomatoes and milk, the three most likely next states capture the semantics of state changes in an accurate manner.

state. The main difference between our approach and these works is that by utilizing relational memory core units we also allow memories to interact with each other during each update.

Perez and Liu (2017) showed that similar ideas can be used to compile supporting memories in tracking dialogue state. Wang et al. (2017) has shown the importance of coreference signals for reading comprehension task. More recently, Dhingra et al. (2018) introduced a specialized recurrent layer which uses coreference annotations for improving reading comprehension tasks. On language modeling task, Ji et al. (2017) proposed a language model which can explicitly incorporate entities while dynamically updating their representations for a variety of tasks such as language modeling, coreference resolution, and entity prediction.

Our work builds upon and contributes to the growing literature on tracking states changes in procedural text. Bosselut et al. (2018) presented a neural model that can learn to explicitly predict state changes of ingredients at different points in a cooking recipe. Dalvi et al. (2018) proposed another entity-aware model to track entity states in scientific processes. Tandon et al. (2018) demonstrated that the prediction quality can be boosted by including hard and soft constraints to eliminate unlikely or favor probable state changes. In a follow-up work, Du et al. (2019) exploited the notion of label consistency in training to enforce similar predictions in similar procedural contexts. Das et al. (2019) proposed a model that dynamically constructs a knowledge graph while reading the procedural text to track the ever-changing entities states. As discussed in the introduction, however, these previous methods use a strong inductive bias and assume that state labels are present during training. In our study, we deliberately focus on unlabeled procedural data and ask the question: Can multimodality help to identify and provide insights to understanding state changes.

6 Conclusion

We have presented a new neural architecture called Procedural Reasoning Networks (PRN) for multimodal understanding of step-by-step instructions. Our proposed model is based on the successful BiDAF framework but also equipped with an explicit memory unit that provides an implicit mecha-

nism to keep track of the changes in the states of the entities over the course of the procedure. Our experimental analysis on visual reasoning tasks in the RecipeQA dataset shows that the model significantly improves the results of the previous models, indicating that it better understands the procedural text and the accompanying images. Additionally, we carefully analyze our results and find that our approach learns meaningful dynamic representations of entities without any entity-level supervision. Although we achieve state-of-the-art results on RecipeQA, clearly there is still room for improvement compared to human performance. We also believe that the PRN architecture will be of value to other visual and textual sequential reasoning tasks.

Acknowledgements

We thank the anonymous reviewers and area chairs for their invaluable feedback. This work was supported by TUBA GEBIP fellowship awarded to E. Erdem; and by the MMVC project via an Institutional Links grant (Project No. 217E054) under the Newton-Katip Çelebi Fund partnership funded by the Scientific and Technological Research Council of Turkey (TUBITAK) and the British Council. We also thank NVIDIA Corporation for the donation of GPUs used in this research.

References

Trapit Bansal, Arvind Neelakantan, and Andrew McCallum. 2017. RelNet: End-to-End Modeling of Entities & Relations. In *NeurIPS Workshop on Automated Knowledge Base Construction (AKBC)*.

Antoine Bosselut, Corin Ennis, Omer Levy, Ari Holtzman, Dieter Fox, and Yejin Choi. 2018. Simulating Action Dynamics with Neural Process Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A Thorough examination of the CNN/Daily Mail Reading Comprehension Task. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2358–2367.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2019. Building Dynamic Knowledge Graphs from Text using Machine Reading Comprehension. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2018. Neural Models for Reasoning over Multiple Mentions using Coreference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Xinya Du, Bhavana Dalvi Mishra, Niket Tandon, Antoine Bosselut, Wen-tau Yih, Peter Clark, and Claire Cardie. 2019. Be consistent! improving procedural text comprehension using label consistency. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Rearning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking The World State with Recurrent Entity Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 1693–1701.

Schmidhuber J. Hochreiter, S. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

Mohit Iyyer, Varun Manjunatha, Anupam Guha, Yogarshi Vyas, Jordan Boyd-Graber, Hal Daumé III, and Larry Davis. 2017. The amazing mysteries of the gutter: Drawing inferences between panels in comic book narratives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic Entity Representations in Neural Language Models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Samira Ebrahimi Kahou, Adam Atkinson, Vincent Michalski, Akos Kadar, Adam Trischler, and Yoshua Bengio. 2017. FigureQA: An Annotated Figure Dataset for Visual Reasoning. In *Proceedings of the International Conference on Learning Representations Workshop (ICLR Workshop)*.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are You Smarter Than A Sixth Grader? Textbook Question Answering for Multimodal Machine Comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2018. Recipe1M: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. *arXiv preprint arXiv:1810.06553*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic Differentiation in pytorch. In *NIPS-W*.

Juan Pavez, Hector Allende, and Hector Allende-Cid. 2018. Working memory networks: Augmenting memory networks with a relational reasoning module. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1000–1009.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using memory network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 305–314.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2227–2237.

Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. 2018. Relational Recurrent Neural Networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 7299–7310.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017a. Bidirectional Attention Flow for Machine Comprehension. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017b. Query-Reduction Networks for Question Answering. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

R. K. Srivastava, K. Greff, and J. Schmidhuber. 2015. Highway networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-To-End Memory Networks. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 2440–2448.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826.

Niket Tandon, Bhavana Dalvi, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. 2018. Reasoning about actions and state changes by injecting commonsense knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. MovieQA: Understanding Stories in Movies Through Question-Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4631–4640.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008.

Hai Wang, Takeshi Onishi, Kevin Gimpel, and David McAllester. 2017. Emergent predication structure in hidden state vectors of neural readers. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 26–36, Vancouver, Canada. Association for Computational Linguistics.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. 2018. RecipeQA: A Challenge Dataset for Multimodal Comprehension of Cooking Recipes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yummly. 2015. Kaggle Whats Cooking? https://www.kaggle.com/c/whats-cooking/data. [Accessed: 2018-05-31].

On the Limits of Learning to Actively Learn Semantic Representations

Omri Koshorek[1] **Gabriel Stanovsky**[2,4] **Yichu Zhou**[3]

Vivek Srikumar[3] **Jonathan Berant**[1,2]

[1]Tel-Aviv University, [2]Allen Institute for AI
[3]The University of Utah, [4]University of Washington
{omri.koshorek,joberant}@cs.tau.ac.il
gabis@allenai.org, {flyaway,svivek}@cs.utah.edu

Abstract

One of the goals of natural language understanding is to develop models that map sentences into meaning representations. However, training such models requires expensive annotation of complex structures, which hinders their adoption. Learning to actively-learn (LTAL) is a recent paradigm for reducing the amount of labeled data by learning a policy that selects which samples should be labeled. In this work, we examine LTAL for learning semantic representations, such as QA-SRL. We show that even an *oracle* policy that is allowed to pick examples that maximize performance on the test set (and constitutes an upper bound on the potential of LTAL), does not substantially improve performance compared to a *random* policy. We investigate factors that could explain this finding and show that a distinguishing characteristic of successful applications of LTAL is the interaction between optimization and the oracle policy selection process. In successful applications of LTAL, the examples selected by the oracle policy do not substantially depend on the optimization procedure, while in our setup the stochastic nature of optimization strongly affects the examples selected by the oracle. We conclude that the current applicability of LTAL for improving data efficiency in learning semantic meaning representations is limited.

1 Introduction

The task of mapping a natural language sentence into a *semantic representation*, that is, a structure that represents its meaning, is one of the core goals of natural language processing. This goal has led to the creation of many general-purpose formalisms for representing the structure of language, such as semantic role labeling (SRL; Palmer et al., 2005), semantic dependencies (SDP; Oepen et al., 2014), abstract meaning representation (AMR; Banarescu et al., 2013), universal conceptual cognitive annotation (UCCA; Abend and Rappoport, 2013), question-answer driven SRL (QA-SRL; He et al., 2015), and universal dependencies (Nivre et al., 2016), as well as domain-specific semantic representations for particular users in fields such as biology (Kim et al., 2009; Nédellec et al., 2013; Berant et al., 2014) and material science (Mysore et al., 2017; Kim et al., 2019).

Currently, the dominant paradigm for building models that predict such representations is supervised learning, which requires annotating thousands of sentences with their correct structured representation, usually by experts. This arduous data collection is the main bottleneck for building parsers for different users in new domains.

Past work has proposed directions for accelerating data collection and improving data efficiency through multi-task learning across different representations (Stanovsky and Dagan, 2018; Hershcovich et al., 2018), or having non-experts annotate sentences in natural language (He et al., 2015, 2016). One of the classic and natural solutions for reducing annotation costs is to use *active learning*, an iterative procedure for selecting unlabeled examples which are most likely to improve the performance of a model, and annotating them (Settles, 2009).

Recently, learning to actively-learn (LTAL) has been proposed (Fang et al., 2017; Bachman et al., 2017; Liu et al., 2018), where the procedure for selecting unlabeled examples is *trained* using methods from reinforcement and imitation learning. In recent work by Liu et al. (2018), given a labeled dataset from some domain, active learning is simulated on this dataset, and a policy is trained to iteratively select the subset of examples that maximizes performance on a development set. Then, this policy is used on a target domain to select unlabeled examples for annotation. If the learned

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 452–462
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

policy generalizes well, we can reduce the cost of learning semantic representations. Liu et al. (2018) and Vu et al. (2019) have shown that such learned policies significantly reduce annotation costs on both text classification and named entity recognition (NER).

In this paper, we examine the potential of LTAL for learning a semantic representation such as QA-SRL. We propose an *oracle setup* that can be considered as an upper bound to what can be achieved with a learned policy. Specifically, we use an *oracle policy* that is allowed to always pick a subset of examples that maximizes its target metric on a development set, which has the same distribution as the *test set*. Surprisingly, we find that even this powerful oracle policy does not substantially improve performance compared to a policy that randomly selects unlabeled examples on two semantic tasks: QA-SRL span (argument) detection and QA-SRL question (role) generation.

To elucidate this surprising finding, we perform a thorough analysis, investigating various factors that could negatively affect the oracle policy selection process. We examine possible explanatory factors including: (a) the search strategy in the unlabeled data space (b) the procedure for training the QA-SRL model (c) the architecture of the model and (d) the greedy nature of the selection procedure. We find that for all factors, it is challenging to get consistent gains with an oracle policy over a random policy.

To further our understanding, we replicate the experiments of Liu et al. (2018) on NER, and compare the properties of a successful oracle policy in NER to the less successful case of QA-SRL. We find that optimization stochasticity negatively affects the process of sample selection in QA-SRL; different random seeds for the optimizer result in different selected samples. We propose a measure for quantifying this effect, which can be used to assess the potential of LTAL in new setups.

To conclude, in this work, we conduct a thorough empirical investigation of LTAL for learning a semantic representation, and find that it is difficult to substantially improve data efficiency compared to standard supervised learning. Thus, other approaches should be explored for the important goal of reducing annotation costs in building such models. Code for reproducing our experiments is available at https://github.com/koomri/LTAL_SR/.

2 Learning to Actively Learn

Classic pool-based active learning (Settles, 2009) assumes access to a small labeled dataset $\mathcal{S}_{\text{lab}}$ and a large pool of unlabeled examples $\mathcal{S}_{\text{unlab}}$ for a target task. In each iteration, a heuristic is used to select L unlabeled examples, which are sent to annotation and added to $\mathcal{S}_{\text{lab}}$. An example heuristic is *uncertainty sampling* (Lewis and Gale, 1994), which at each iteration chooses examples that the current model is the least confident about.

LTAL proposes to replace the heuristic with a learned policy π_θ, parameterized by θ. At *training time*, the policy is trained by simulating active learning on a labeled dataset and generating training data from the simulation. At *test time*, the policy is applied to select examples in a new domain. Figure 1 and Algorithm 1 describe this data collection procedure, on which we build our *oracle policy* (§3).

In LTAL, we assume a labeled dataset $\mathcal{D}$ which is partitioned into three disjoint sets: a small labeled set $\mathcal{S}_{\text{lab}}$, a large set $\mathcal{S}_{\text{unlab}}$ that will be treated as unlabeled, and an evaluation set $\mathcal{S}_{\text{eval}}$ that will be used to estimate the quality of models. Then, active learning is simulated for B iterations. In each iteration i, a model m^i_ϕ, parameterized by ϕ, is first trained on the labeled dataset. Then, K subsets $\{\mathcal{C}_j\}_{j=1}^K$ are randomly sampled from $\mathcal{S}_{\text{unlab}}$, and the model m^i_ϕ is fine-tuned on each candidate set, producing K models $\{m^i_{\phi_j}\}_{j=1}^K$. The performance of each model is evaluated on $\mathcal{S}_{\text{eval}}$, yielding the scores $\{s(\mathcal{C}_j)\}_{j=1}^K$. Let the candidate set with highest accuracy be $\mathcal{C}^i_t$. We can create training examples for π_θ, where $(\mathcal{S}_{\text{lab}}, \mathcal{S}_{\text{unlab}}, m^i_\phi, \{s(\mathcal{C}_j)\}_{j=1}^K)$ are the inputs and $\mathcal{C}^i_t$ is the label. Then $\mathcal{C}^i_t$ is moved from $\mathcal{S}_{\text{unlab}}$ to $\mathcal{S}_{\text{lab}}$.

Simulating active learning is a computationally expensive procedure. In each iteration we need to train K models over $\mathcal{S}_{\text{lab}} \cup \mathcal{C}_j$. However, a trained network can potentially lead to a policy that is better than standard active learning heuristics.

3 An Oracle Active Learning Policy

Our goal is to examine the potential of LTAL for learning a semantic representation such as QA-SRL. Towards this goal, we investigate an *oracle policy* that should be an upper bound for what can be achieved with a learned policy π_θ.

The oracle policy is allowed to use Algorithm 1

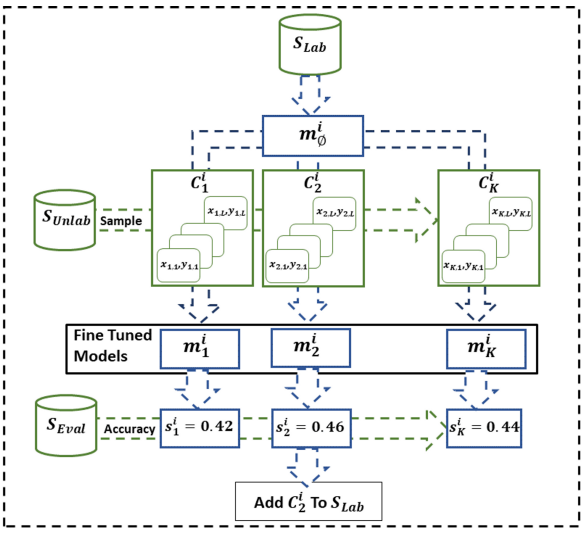


Figure 1: A single iteration of LTAL, where examples are sampled from $\mathcal{S}_{\text{unlab}}$, trained with examples in $\mathcal{S}_{\text{lab}}$, and performance on $\mathcal{S}_{\text{eval}}$ is used to select examples to annotate. See §2 for details.

Algorithm 1: Simulating active learning

Input: $\mathcal{S}_{\text{lab}}, \mathcal{S}_{\text{unlab}}, \mathcal{S}_{\text{eval}}$

1 **for** $i \in \{1 \dots B\}$ **do**
2 $\quad m^i_\phi \leftarrow \text{Train}(\mathcal{S}_{\text{lab}})$
3 $\quad \mathcal{C}^i_1, \dots, \mathcal{C}^i_K = \text{SampleCandidates}(\mathcal{S}_{\text{unlab}})$
4 $\quad$ **for** $j \in \{1, \dots, K\}$ **do**
5 $\quad\quad m^i_{\phi_j} \leftarrow \text{FineTune}(m^i_\phi, \mathcal{S}_{\text{lab}} \cup \mathcal{C}^i_j)$
6 $\quad\quad s^i_j \leftarrow \text{Accuracy}(m^i_{\phi_j}, \mathcal{S}_{\text{eval}})$
7 $\quad t \leftarrow \underset{j \in \{1,\dots,K\}}{\text{argmax}}\ s^i_j$
8 $\quad \text{CreateTrainEx}((\mathcal{S}_{\text{lab}}, \mathcal{S}_{\text{unlab}}, m^i_\phi, \{\mathcal{C}^i_j\}^K_{j=1}), \mathcal{C}^i_t)$
9 $\quad \mathcal{S}_{\text{lab}} \leftarrow \mathcal{S}_{\text{lab}} \cup \mathcal{C}^i_t, \mathcal{S}_{\text{unlab}} \leftarrow \mathcal{S}_{\text{unlab}} \setminus \mathcal{C}^i_t$
10 **return** $\mathcal{S}_{\text{lab}}$

at test time (it does not create training examples for π_θ, thus Line 8 is skipped). Put differently, the oracle policy selects the set of unlabeled examples that maximizes the target metric of our model on a set sampled from the *same distribution* as the test set. Therefore, the oracle policy enjoys extremely favorable conditions compared to a trained policy, and we expect it to provide an upper bound on the performance of π_θ. Despite these clear advantages, we will show that an oracle policy struggles to substantially improve performance compared to a random policy.

While the oracle policy effectively "peeks" at the label to make a decision, there are various factors that could explain the low performance of a model trained under the oracle policy. We now list several hypotheses, and in §5.4 and §6 methodologically examine whether they explain the empirical results of LTAL.

- **Training**: The models $m^i_{\phi_j}$ are affected by the training procedure in Lines 2 and 5 of Alg. 1. Different training procedures affect the performance of models trained with the oracle policy.
- **Search space coverage**: Training over all unlabeled examples in each iteration is intractable, so the oracle policy randomly samples K subsets, each with L examples. Because $K \cdot L << |\mathcal{S}_{\text{unlab}}|$, it is possible that randomly sampling these sets will miss the more beneficial unlabeled examples. Moreover, the parameter L controls the diversity of candidate subsets, since as L increases the similarity between the K different subsets grows. Thus, the hyper-parameters K and L might affect the outcome of the oracle policy.
- **Model architecture**: The model architecture (e.g., number of parameters) can affect the efficacy of learning under the oracle policy.
- **Stochasticity:** The oracle policy chooses an unlabeled set based on performance after training with stochastic gradient descent. Differences in performance between candidate sets might be related to this stochasticity, rather than to the actual value of the examples (especially when $\mathcal{S}_{\text{lab}}$ is small).
- **Myopicity:** The oracle policy chooses the set $\mathcal{C}^i_j$ that maximizes its performance. However, the success of LTAL depends on the *sequence* of choices that are made. It is possible that the greedy nature of this procedure results in suboptimal performance. Unfortunately, improving search through beam search or similar measures is intractable in this already computationally-expensive procedure.

We now describe QA-SRL (He et al., 2015), which is the focus of our investigation, and then describe the experiments with the oracle policy.

4 QA-SRL Schema

QA-SRL was introduced by He et al. (2015) as an open variant of the predefined role schema in traditional SRL. QA-SRL replaces the predefined set of *roles* with the notion of *argument questions*. These are natural language questions centered around a target predicate, where the answers to the given question are its corresponding arguments. For example, for the sentence *"Elizabeth Warren decided to **run** for president"*, traditional SRL will label *"Elizabeth Warren"* as `ARG0` of the **run** predicate (the agent of the predicate, or the entity running in this case), while QA-SRL

*Elizabeth Warren **announced** her candidacy at a rally in Massachusetts.*

Argument	QA-SRL role	PropBank role
Elizabeth Warren	Who **announced** something?	ARG0
her candidacy	What did someone **announce**?	ARG1
at a rally in Massachusetts	Where did someone **announce** something?	ARGM-LOC

Table 1: Example of QA-SRL versus traditional SRL annotation for a given input sentence (top). Each line shows a single argument, and its role in QA-SRL (in question form) followed by its traditional SRL role, using PropBank notation. Roles in QA-SRL have a structured open representation, while SRL assigns discrete roles from a predefined set.

will assign the more subtle question *"who might run?"*, indicating the uncertainty of this future event. Questions are generated by assigning values to 7 pre-defined slots (where some of the slots are potentially empty). See Table 1 for an example QA-SRL annotation of a full sentence.

Recently, FitzGerald et al. (2018) demonstrated the scalability of QA-SRL by crowdsourcing the annotation of a large QA-SRL dataset, dubbed QA-SRL bank 2.0. It consists of 250K QA pairs over 64K sentences on three different domains (Wikipedia, news, and science). Following, this large dataset has enabled the development a neural model which breaks QA-SRL into a pipeline of two tasks, given a target predicate in an input sentence. First, a *span detection* algorithm identifies arguments of the predicate as continuous spans in the sentence (e.g., *"Elizabeth Warren"* in the previous example), then a *question generation* model predicts an appropriate role question (e.g., *"who might run?"*).

We find that QA-SRL is a good test-bed for active learning of semantic representations, for several key reasons: (1) it requires semantic understanding of the sentence, beyond syntactic or surface-level features (e.g., identifying the factuality of a given predicate), (2) adopting the formulation of FitzGerald et al. (2018), it consists of two semantic tasks, allowing us to test active learning on both of them, (3) we can leverage the large QA-SRL dataset to simulate active learning scenarios, and lastly (4) QA-SRL's scalability is attractive for the application of active learning policies, as they may further reduce costs for researchers working on developing specialized semantic representations in low-resource domains (e.g., medical, biological, or educational domains).

5 Experimental Evaluation

We now perform a series of experiments comparing the performance of an oracle policy to a random policy. We describe the experimental settings (§5.1), tasks and models (§5.2), present the main results (§5.3), and conclude by investigating factors that might affect our empirical findings (§5.4).

5.1 Experimental Settings

We evaluate the potential of the oracle policy on QA-SRL Bank 2.0 (FitzGerald et al., 2018). We use the training set of the *science* domain as $\mathcal{D}$, randomly split it into $\mathcal{S}_{\text{lab}}$, $\mathcal{S}_{\text{unlab}}$, and $\mathcal{S}_{\text{eval}}$. We evaluate the success of a model m^i_ϕ trained with the oracle policy by periodically measuring performance on the development set of the *science* domain. Unless mentioned, all results are an average of 3 experiments, where a different split of $\mathcal{D}$ was performed. Each experiment used K threads of a 40-core 2.2GHz Xeon Silver 4114 machine.

We compare the results of a base oracle policy (BASEORACLE) corresponding to the best policy we were able to obtain using the architecture from FitzGerald et al. (2018) to the following baselines:

- RANDOM: One of the candidate sets $\mathcal{C}^i_j$ is chosen at random and added to $\mathcal{S}_{\text{lab}}$.
- LONGEST: The set $\mathcal{C}^i_j$ with the maximal average number of tokens per sentence is added to $\mathcal{S}_{\text{lab}}$.
- UNCERTAINTY: For each candidate set, we use m^i_ϕ to perform predictions over all of the sentences in the set, and choose the set $\mathcal{C}^i_j$ that has the maximal average entropy over the set of predictions.

5.2 Tasks and Models

We now describe the three tasks and corresponding models in our analysis:

Span Detection: Here we detect spans that are arguments of a predicate in a sentence (see Table 1). We start with a labeled set of size $|\mathcal{S}_{\text{lab}}| = 50$, and select examples with the oracle policy for $B = 460$ iterations. We set the number of candidate sets to $K = 5$, and the size of each set to $L = 1$,

thus the size of the final labeled set is 510 examples. We train the publicly available span detection model released by FitzGerald et al. (2018), which consumes as input a sentence $x_1, \ldots, x_n$, where x_i is the concatenation of the embedding of the ith word in the sentence and a learned embedding of a binary indicator for whether this word is the target predicate. This input is fed into a multi-layer encoder, producing a representation h_i for every token. Each span $x_{i:j}$ is represented by concatenating the respective hidden states: $s_{ij} = [h_i; h_j]$. A fully connected network consumes the span representation s_{ij}, and predicts a probability whether the span is an argument or not.

To accelerate training, we reduce the number of parameters to 488K by freezing the token embeddings, reducing the number of layers in the encoder, and by shrinking the dimension of both the hidden representations and the binary predicate indicator embedding. Following FitzGerald et al. (2018), we use GLoVe embeddings (Pennington et al., 2014).

Question Generation: We generate the question (role) for a given predicate and corresponding argument. We start with a labeled set of size $|\mathcal{S}_{\text{lab}}| = 500$ and perform $B = 250$ iterations, where in each iteration we sample $K = 5$ candidate sets each of size $L = 10$ (lower values were intractable). Thus, the final size of $\mathcal{S}_{\text{lab}}$ is 3,000 samples. We train the publicly available *local* question generation model from FitzGerald et al. (2018), where the learned argument representation s_{ij} is used to independently predict each of the 7 question slots. We reduce the number of parameters to 360K with the same modifications as in the span detector model. As a metric for the quality of question generation models, we use its official metric exact match (EM), which reflects the percentage of predicted questions that are identical to the ground truth questions.

Named Entity Recognition: To reproduce the experiments of Liu et al. (2018) we run the oracle policy on the CoNLL-2003 NER English dataset (Sang and De Meulder, 2003), replicating the experimental settings described in Liu et al. (2018) (as their code is not publicly available). We run the oracle policy for $B = 200$ iterations, starting from an empty $\mathcal{S}_{\text{lab}}$, and adding one example ($L = 1$) from $K = 5$ candidate sets in each iteration. We use a CRF sequence tagger from AllenNLP (Gardner et al., 2018), and experiment with two variants: (1) NER-MULTILANG: A BiLSTM CRF model (20K parameters) with 40 dimensional multi-lingual word embeddings (Ammar et al., 2016), and (2) NER-LINEAR: A linear CRF model which was originally used by Liu et al. (2018).

5.3 Results

Span Detection: Table 2 shows F_1 score (the official metric) of the QA-SRL span detector models for different sizes of $\mathcal{S}_{\text{lab}}$ for BASEORACLE and the other baselines. Figure 2 (left) shows the relative improvement of the baselines over RANDOM. We observe that the maximal improvement of BASEORACLE over RANDOM is 9% given 200 examples, but with larger $\mathcal{S}_{\text{lab}}$ the improvement drops to less than 5%. This is substantially less than the improvements obtained by Liu et al. (2018) on text classification and NER. Moreover, LONGEST outperforms BASEORACLE in most of the observed results. This shows that there exists a selection strategy that is better than BASEORACLE, but it is not the one chosen by the oracle policy.

Question Generation: To check whether the previous result is specific to span detection, we conduct the same experiment for question generation. However, training question generation models is slower compared to span detection and thus we explore a smaller space of hyper-parameters. Table 3 reports the EM scores achieved by BASEORACLE and LONGEST, and Figure 2 (center) shows the relative improvement. Here, the performance of BASEORACLE is even worse compared to span detection, as its maximal relative improvement over RANDOM is at most 5%.

Named Entity Recognition: Figure 2 (right) shows the relative improvement of NER-LINEAR and NER-MULTILANG compared to RANDOM. We observe that in NER-LINEAR, which is a replication of Liu et al. (2018), the oracle policy indeed obtains a large improvement over RANDOM for various sizes of $\mathcal{S}_{\text{lab}}$, with at least 9.5% relative improvement in performance. However, in NER-MULTILANG the relative gains are smaller, especially when the size of $\mathcal{S}_{\text{lab}}$ is small.

5.4 Extended Experiments

Surprisingly, we observed in §5.3 that even an oracle policy, which is allowed to pick the examples

# samples	100	150	200	250	300	350	400	450	500
BASEORACLE	42.7	**49.2**	52.9	54.2	**56.6**	**57.4**	58.4	**59.5**	59.9
RANDOM	42.8	47.2	48.3	52.4	53.3	56.1	57.0	57.5	58.5
LONGEST	**44.1**	49.1	**53.0**	**55.5**	56.4	**57.4**	**58.7**	58.6	**60.0**
UNCERTAINTY	42.8	47.0	50.1	51.3	52.2	54.4	55.1	55.6	56.9

Table 2: Span detection F_1 on the development set for all models across different numbers of labeled examples.

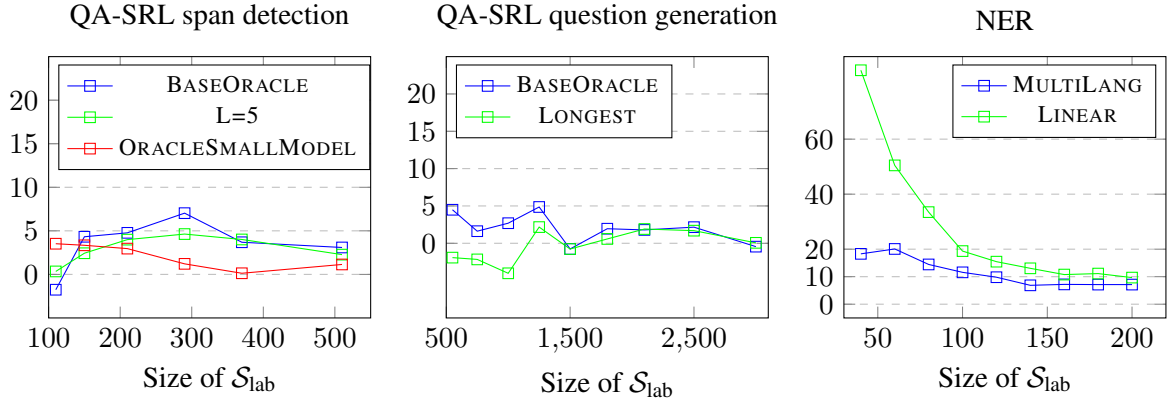


Figure 2: Relative improvement (in %) of different models compared to RANDOM on the development set. Note that the range of the y-axis in NER is different from QA-SRL.

that maximize performance on samples from the same distribution as the test set, does not substantially improve performance over a random policy. One possibility is that no active learning policy is better than random. However, LONGEST outperformed BASEORACLE showing that the problem is at least partially related to BASEORACLE itself.

We now examine the possible factors described in §3 and investigate their interaction with the performance of models trained with BASEORACLE. All modifications were tested on *span detection*, using the experimental settings described in §5.1.

Search space coverage We begin by examining the effect of the parameters K and L on the oracle policy. As K increases, we cover more of the unlabeled data, but training time increases linearly. As L increases, the subsets $\{\mathcal{C}_j\}_{j=1}^{K}$ become more similar to one another due to the fact that we are randomly mixing more examples from the unlabeled data. On the other hand, when L is small, the fine-tuning process is less affected by the candidate sets and more by $\mathcal{S}_{\text{lab}}$. In such case, it is likely that the difference in scores is also affected by stochasticity.

BASEORACLE uses $K = 5, L = 1$. We examine the performance of the oracle policy as these values are increased in Table 4. We observe that performance does not improve, and perhaps even decreases for larger values of K. We hypothesize that a large K increases the greediness of the procedure, and may result in selecting an example that seems promising in the current iteration but is suboptimal in the long run, similar to large beam sizes reducing performance in neural machine translation (Yang et al., 2018). A moderate K results in a more random and possibly beneficial selection.

Increasing the size of each candidate set to $L = 5$ or 20 results in roughly similar performance to $L = 1$. We hypothesize that there is a trade-off where as L increases the similarity between the different sets increases but training becomes more stable and vice versa, and thus performance for different L values does not vary substantially.

Training In Lines 2 and 5 of Alg. 1 we train on $\mathcal{S}_{\text{lab}}$ and then fine-tune on the union $\mathcal{S}_{\text{lab}} \cup \mathcal{C}_j^i$ until s_j^i does not significantly improve for 5 epochs. It is possible that fine-tuning from a fixed model reduces the efficacy of training, and training on $\mathcal{S}_{\text{lab}} \cup \mathcal{C}_j^i$ from random weights will improve performance. Of course, training from scratch will substantially increase training time. We run an experiment, termed INDEP., where Line 2 is skipped, and in Line 5 we independently train each of the candidate models from random weights. We find that this modification does not achieve better results than BASEORACLE, possibly because training a model from scratch for each of the candidates increases the stochasticity in the optimization.

# samples	550	750	1000	1250	1500	1800	2100	2500	3000
BASEORACLE	**18.9**	**21.7**	**24.4**	**26.4**	27.1	**28.4**	**29.1**	**30.6**	31.1
RANDOM	18.1	21.4	23.7	25.2	**27.3**	27.9	28.6	29.9	**31.3**
LONGEST	17.8	20.9	22.8	25.7	27.1	28.0	**29.1**	30.4	**31.3**

Table 3: Question generation scores (exact match) on the development set across different numbers of labeled examples.

# samples	110	150	210	290	370	510
RANDOM	45.2	47.2	50.5	53.1	55.8	58.5
BASEORACLE	43.3	**49.2**	**52.9**	**56.8**	57.8	**60.3**
$K = 10$	**46.6**	48.8	51.4	55.8	57.6	58.6
$K = 20$	44.8	47.4	52.1	55.9	—	—
$L = 5$	44.2	48.3	52.5	55.5	**58.0**	59.8
$L = 20$	45.2	47.5	51.9	55.1	56.7	58.7
LOSS-SCORE	30.0	38.2	41.0	51.5	53.7	57.2
INDEP.*	40.9	44.6	50.1	54.1	—	—
EPSILON-GREEDY-0.3	45.1	48.6	52.4	55.6	57.1	59.8
ORACLE-100	44.9	48.8	51.4	53.9	57.0	59.2
RANDOMSMALLMODEL	51.9	54.8	57.3	59.5	61.4	62.6
ORACLESMALLMODEL	53.8	56.6	58.9	60.3	61.5	63.3

Table 4: Span detection F_1 scores on the development set for different size of $\mathcal{S}_{\text{lab}}$. We highlight the best performing policy for the standard span detector architecture. (*) indicates that the results are for a single run.

In addition, we also experiment with fine-tuning on $\mathcal{C}_j$ only, rather than $\mathcal{S}_{\text{lab}} \cup \mathcal{C}_j$. As we expect, results are quite poor since the model uses only a few examples for fine-tuning and forgets the examples in the labeled set.

Lastly, we hypothesize that selecting a candidate set based on the target metric (F_1 for span detection) might not be sensitive enough and thus we run an experiment, termed LOSS-SCORE, where we select the set $\mathcal{C}_j$ that minimizes the loss on the development set. We find that this modification achieves lower results than RANDOM, especially when $\mathcal{S}_{\text{lab}}$ is small, reflecting the fact that the loss is not perfectly correlated with our target metric.

Model Architecture In §5.3 we observed that results on NER vary with the model architecture. To see whether this phenomenon occurs also for span detection we perform a modification to the model – we reduce the number of parameters from 488K to 26K by reducing the hidden state size and replacing GLoVe embeddings with multi-lingual embeddings (Ammar et al., 2016). We then compare an oracle policy (ORACLESMALLMODEL) with a random policy (RANDOMSMALLMODEL). Table 4 shows that while absolute F_1 actually improves in this setup, the oracle policy improves performance compared to a random policy by no more than 4%. Thus, contrary to NER, here architecture modifications do not expose an advantage of the oracle policy compared to the random one. We did not examine a simpler linear model for span detection, in light of recent findings (Lowell et al., 2019) that it is important to test LTAL with state-of-the-art models, as performance is tied to the specific model being trained.

Myopicity We hypothesized that greedily selecting an example that maximizes performance in a specific iteration might be suboptimal in the long run. Because non-greedy selection strategies are computationaly intractable, we perform the following two experiments.

First, we examine EPSILON-GREEDY-P, where in each iteration the oracle policy selects the set $\mathcal{C}_j$ that maximizes target performance with probability $1 - p$ and randomly chooses a set with probability p. This is meant to check whether adding random exploration to the oracle policy might prevent it from getting stuck in local optima. We find that when $p = 0.3$ its performance is comparable to BASEORACLE while reducing the computational costs.

Second, we observe that most of the gain of BASEORACLE compared to RANDOM is in the beginning of the procedure. Thus, we propose to use BASEORACLE in the first b iterations, and then transition to a random policy (termed ORACLE-B). We run this variation with $b = 100$ and find that it leads to similar performance.

To summarize, we have found that an oracle policy only slightly improves performance for QA-SRL span detection and question generation compared to a random policy, and that improvements in NER are also conditioned on the underlying model. Our results echo recent findings by Lowell et al. (2019), who have shown that gains achieved by active learning are small and inconsistent when modifying the model architecture.

We have examined multiple factors that might affect the performance of models trained with an oracle policy including the training procedure, model architecture, and search procedure, and have shown that in all of them the oracle policy

struggles to improve over the random one. Thus, a *learned policy* is even less likely to obtain meaningful gains using LTAL.

In the next section we analyze the differences between NER-LINEAR, where LTAL works well, and BASEORACLE, in order to better understand the underlying causes for this phenomenon.

6 When does LTAL Work?

A basic underlying assumption of active learning (with or without a learned policy), is that some samples in $\mathcal{S}_{\text{unlab}}$ are more informative for the learning process than others. In LTAL, the informativeness of a candidate example set is defined by the accuracy of a trained model, as evaluated on $\mathcal{S}_{\text{eval}}$ (Line 6 in Alg. 1). Thus, for active learning to work, the candidate set that is selected should not be affected by the stochasticity of the training process. Put differently, the ranking of the candidate sets by the oracle policy should be *consistent* and not be dramatically affected by the optimization.

To operationalize this intuition, we use Alg. 1, but run the for-loop in Line 4 twice, using two different random seeds. Let $\mathcal{C}_t^i$ be the chosen or *reference* candidate set according to the first run of the for-loop in iteration i. We can measure the consistency of the optimization process by looking at the ranking of the candidate sets $\mathcal{C}_1^i, \ldots \mathcal{C}_K^i$ according to the second fine-tuning, and computing the mean reciprocal rank (MRR) with respect to the reference candidate set $\mathcal{C}_t^i$ across all iterations:

$$\text{MRR} = \frac{1}{B}\sum_{i=1}^{B}\frac{1}{\text{rank}(\mathcal{C}_t^i)}, \qquad (1)$$

where $\text{rank}(\mathcal{C}_t^i)$ is the rank of $\mathcal{C}_t^i$ in the second fine tuning step. The only difference between the two fine-tuning procedures is the random seed. Therefore, an MRR value that is close to 1 means that the ranking of the candidates is mostly affected by the quality of the samples, while a small MRR hints that optimization plays a large role. We prefer MRR to other correlation-based measures (such as Spearman's rank-order correlation), because the oracle is only affected by the candidate set that is ranked first. We can now examine whether the MRR score correlates with whether LTAL works or not.

We measure the MRR in 3 settings: (1) NER-LINEAR, a linear CRF model for NER which replicates the experimental settings in (Liu et al., 2018), where LTAL works, (2) NER-MULTILANG, a BiLSTM-CRF sequence tagger from AllenNLP (Gardner et al., 2018) with 40 dimensional multi-lingual word embeddings of Ammar et al. (2016), and (3) BASEORACLE, the baseline model for span detection task. In all experiments the initial $\mathcal{S}_{\text{lab}}$ was empty and $B = 200$, following the experimental settings in which LTAL has shown good performance (Liu et al., 2018; Fang et al., 2017; Vu et al., 2019). Since the MRR might change as the size of $\mathcal{S}_{\text{lab}}$ is increasing, we compute and report MRR every 10 iterations.

Figure 3 (left) presents the MRR in the three experiments. We observe that in NER-LINEAR the MRR has a stable value of 1, while in NER-MULTILANG and BASEORACLE the MRR value is substantially lower, and closer to an MRR value of a random selection (~.46). The right side of Figure 3 shows that NER-LINEAR oracle policy outperforms a random policy by a much larger margin, compared to the other 2 experiments.

These results show that the ranking in NER-LINEAR is not affected by the stochasticity of optimization, which is expected given its underlying convex loss function. On the other hand, the optimization process in the other experiments is over a non-convex loss function and a small $\mathcal{S}_{\text{lab}}$, and thus optimization is more brittle. Interestingly, we observe in Figure 3 that the gains of the oracle policy in NER-LINEAR are higher than NER-MULTILANG, although the task and the dataset are exactly same in the two experiments. This shows that the potential of LTAL is affected by the model, where a more complex model leads to smaller gains by LTAL.

We view our findings as a guideline for future work: by tracking the MRR one can assess the potential of LTAL at development time – when the MRR is small, the potential is limited.

7 Related Work

Active learning has shown promising results on various tasks. The commonly used *uncertainty* criteria (Lewis and Catlett, 1994; Culotta and McCallum, 2005) is focused on selecting the samples on which the confidence of the model is low. Among other notable approaches, in *query by committee* (Seung et al., 1992) a disagreement between a set of trained models on the prediction of an example is used to select what samples to label.

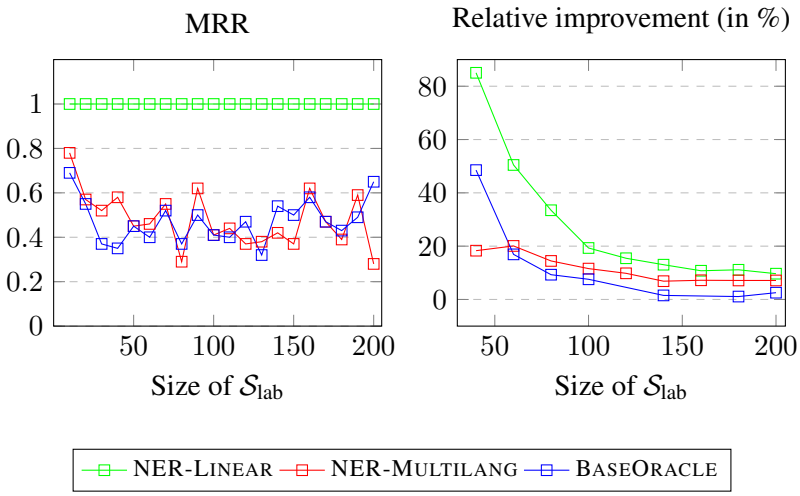


Figure 3: MRR (on the left) and relative improvement (in %) of different models compared to RANDOM on the development set.

In a large empirical study, Lowell et al. (2019) have recently shown other limitations in active learning. They investigate the performance of active learning across NLP tasks and model architectures, and demonstrate that it does not achieve consistent gains over supervised learning, mostly because the collected samples are beneficial to a specific model architecture, and does not yield better results than random selection when switching to a new architecture.

There has been little research regarding active learning of semantic representations. Among the relevant work, Siddhant and Lipton (2018) have shown that *uncertainty* estimation using dropout and Bayes-By-Backprop (Blundell et al., 2015) achieves good results on the SRL formulation. The improvements in performance due to LTAL approaches on various tasks (Konyushkova et al., 2017; Bachman et al., 2017; Fang et al., 2017; Liu et al., 2018) has raised the question whether learned policies can be applied also to the field of learning semantic representations.

8 Conclusions

We presented the first experimentation with LTAL techniques in learning parsers for semantic representations. Surprisingly, we find that LTAL, a learned method which was shown to be effective for NER and document classification, does not do significantly better than a random selection on two semantic representation tasks within the QA-SRL framework, even when given extremely favourable conditions. We thoroughly analyze the factors leading to this poor performance, and find that the stochasticity in the model optimization negatively affects the performance of LTAL. Finally, we propose a metric which can serve as an indicator for whether LTAL will fare well for a given dataset and model. Our results suggest that different approaches should be explored for the important task of building semantic representation models.

Acknowledgements

We thank Julian Michael and Oz Anani for their useful comments and feedback. This research was supported by The U.S-Israel Binational Science Foundation grant 2016257, its associated NSF grant 1737230 and The Yandex Initiative for Machine Learning.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 228–238.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Philip Bachman, Alessandro Sordoni, and Adam Trischler. 2017. Learning algorithms for active learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 301–310. JMLR. org.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin

Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. *EMNLP*.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-scale qa-srl parsing. *arXiv preprint arXiv:1805.05377*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 643–653.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016. Human-in-the-loop parsing. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *Proc. of ACL*, pages 373–385.

Edward Kim, Zach Jensen, Alexander van Grootel, Kevin Huang, Matthew Staib, Sheshera Mysore, Haw-Shiuan Chang, Emma Strubell, Andrew McCallum, Stefanie Jegelka, and Elsa Olivetti. 2019. Inorganic materials synthesis planning with literature-trained neural networks. *CoRR*, abs/1901.00032.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*. Association for Computational Linguistics.

Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. 2017. Learning active learning from data. In *Advances in Neural Information Processing Systems*, pages 4225–4235.

David D Lewis and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR*.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1874–1883.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. 2019. Practical obstacles to deploying active learning. In *Proceedings of EMNLP*.

Sheshera Mysore, Edward Kim, Emma Strubell, Ao Liu, Haw-Shiuan Chang, Srikrishna Kompella, Kevin Huang, Andrew McCallum, and Elsa Olivetti. 2017. Automatically extracting action graphs from materials science synthesis procedures. *CoRR*, abs/1711.06872.

Claire Nédellec, Robert Bossy, Jin-Dong Kim, Jung-Jae Kim, Tomoko Ohta, Sampo Pyysalo, and Pierre Zweigenbaum. 2013. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pages 1–7.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: Broad-coverage semantic dependency parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 63–72.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Burr Settles. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM.

Aditya Siddhant and Zachary C Lipton. 2018. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. *arXiv preprint arXiv:1808.05697*.

Gabriel Stanovsky and Ido Dagan. 2018. Semantics as a foreign langauge. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium. Association for Computational Linguistics.

Thuy Vu, Ming Liu, Dinh Phung, and Gholamreza Haffari. 2019. Learning how to active learn by dreaming. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4091–4101.

Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the beam search curse: A study of (re-) scoring methods and stopping criteria for neural machine translation. *EMNLP*.

How does Grammatical Gender Affect Noun Representations in Gender-Marking Languages?

Hila Gonen[1] **Yova Kementchedjhieva**[2] **Yoav Goldberg**[1,3]
[1]Department of Computer Science, Bar-Ilan University
[2]University of Copenhagen
[3]Allen Institute for Artificial Intelligence
hilagnn@gmail.com,yova@di.ku.dk,yoav.goldberg@gmail.com

Abstract

Many natural languages assign grammatical gender also to inanimate nouns in the language. In such languages, words that relate to the gender-marked nouns are inflected to agree with the noun's gender. We show that this affects the word representations of inanimate nouns, resulting in nouns with the same gender being closer to each other than nouns with different gender. While "embedding debiasing" methods fail to remove the effect, we demonstrate that a careful application of methods that neutralize grammatical gender signals from the words' context when training word embeddings is effective in removing it. Fixing the grammatical gender bias yields a positive effect on the quality of the resulting word embeddings, both in monolingual and cross-lingual settings. We note that successfully removing gender signals, while achievable, is not trivial to do and that a language-specific morphological analyzer, together with careful usage of it, are essential for achieving good results.

1 Introduction

Work on distributional word embeddings focuses almost exclusively on English, or on cross-lingual and language-agnostic techniques. However, languages are diverse and different languages exhibit different linguistic phenomena, which may interact with the English-centric embedding learning algorithms. In this work we look into one such phenomenon—grammatical gender—and examine its effect on the learned representation.

Many languages have rich grammatical systems, that often include a complex gender system as well (Corbett, 1991). Languages with grammatical gender assign and morphologically mark gender not only to animate nouns (which have biological sex, e.g. man, woman, mother, father), but also to inanimate nouns (e.g. dream, book). This grammatical gender assignment is mostly arbitrary: the same inanimate concept can have different gender in different languages. For example, a *flower* is masculine in Italian (*fiore*) and feminine in German (*Blume*).

Languages often maintain an *agreement system* in which certain words agree on different morphological features with other words they relate to. For example, English present-tense verbs are inflected to agree with their nominal subject on the *number* feature. In other languages the agreement system is more elaborate, and in particular verbs, adjectives, determiners and other functions agree with nouns on many features, including gender (Corbett, 2006).[1]

Such grammatical agreement affects the distributional environment of nouns, as nouns of different gender become surrounded by different word forms: feminine nouns co-occur more with the feminine forms of words, while masculine nouns with the masculine forms. For example, the Italian word *viaggio* ("*journey*"-masc) will co-occur with *durato* ("*last*"-masc) and *lungo* ("*long*"-masc), while the word *gita* ("*trip*"-fem) will co-occur with *durata* ("*last*"-fem) and *lunga* ("*long*"-fem).

Such changes in the distributional environment may bias the learned distributional representations of inanimate nouns. Indeed, we see that the majority of the top-10 nearest neighbors of the word *gita* in Italian ("*trip*"-fem) are feminine words. Also, we notice that the word *viaggio* ("*journey*"-masc) is not on the list, while in English, for comparison, we can find *journey* in the top-10 nearest neighbors of *trip*.

In this work, we are interested in investigating, demonstrating and quantifying this effect beyond

[1]As the gender of nouns is fixed, the other elements are inflected to accommodate the agreement constraint. The nouns are said to *assign gender* to the other words.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 463–471
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

the anecdotal level. We also explore methods for removing such unwanted biases.

We demonstrate that both in Italian and in German, the grammatical gender affects similarities between word representations (using words from SimLex-999 (Hill et al., 2015; Leviant and Reichart, 2015)): pairs of nouns with similar gender are closer to each other while pairs of nouns with different gender are farther apart.

After quantifying the effect, we explore several methods of reducing it. A popular choice would be to simply lemmatize all the words prior to feeding them to the embedding learning algorithm. However, full lemmatization can be destructive, in the sense that it will also remove morphological distinction that we may want to keep. We thus seek more surgical approaches. Interestingly, recent embedding debiasing approaches (Bolukbasi et al., 2016) do not work well. We instead look for methods that attempt to neutralize the gender signals from the training data. We find that such methods are effective in reducing the effect, but are also language specific and tricky to get right: we rely on language specific morphological analyzers while carefully accounting for their peculiarities and adjusting our use for each language. We take this work as a reminder that (a) linguistic resources such as lexicons and morphological analyzers are still relevant and useful (cf. (Zalmout and Habash, 2017)); (b) languages are diverse and different languages require different treatments; and (c) small details may matter a lot. In particular, existing tools and resources, either learned or human curated, should not be trusted blindly, but be carefuly adapted for the problem.

Finally, we show that reducing the effect of grammatical agreement also has a positive effect on the quality of the resulting word representations, both in monolingual and cross-lingual settings. We conclude that grammatical gender indeed has its imprints on the representations of inanimate nouns, and that this should be taken into account when working with gender-marking languages. Our code and debiased embeddings are available at `https://github.com/gonenhila/grammatical_gender`.

2 Background and Related Work

Word Embeddings Word embeddings have become an important component in many NLP models and are widely used for a vast range of downstream tasks. These models are based on the distributional hypothesis according to which words that occur in the same contexts tend to have similar meanings (Harris, 1954). Indeed, they aim to create word representations that are derived from their shared contexts, where the context of a word is essentially the words in its proximity (be it according to linear order in the sentence or according to syntactic relations) (Mikolov et al., 2013; Pennington et al., 2014; Levy and Goldberg, 2014).

Gender Biases in Word Embeddings Social gender bias was demonstrated to be consistent and pervasive across different word embeddings (Caliskan et al., 2017). Bolukbasi et al. (2016) show that using word embeddings for simple analogies surfaces many gender stereotypes. In addition, they define the gender bias of a word w by its projection on the "gender direction": $\overrightarrow{w} \cdot (\overrightarrow{he} - \overrightarrow{she})$, assuming all vectors are normalized. Positive bias stands for male-bias. For example, the bias of *manager* is 0.06, while the bias of *nurse* is -0.10[2].

Recently, some work has been done to reduce social gender bias in word embeddings, both as a post-processing step (Bolukbasi et al., 2016) and as part of the training procedure (Zhao et al., 2018). Bolukbasi et al. (2016) use a post-processing debiasing method. Given a word embedding matrix, they make changes to the word vectors in order to reduce the gender bias for all words that are not inherently gendered. They do that by zeroing the gender projection of each word on a predefined gender direction.[3]

In Zmigrod et al. (2019), the authors mitigate social gender bias in gender marking languages using counterfactual data augmentation. Gender-marking languages add several interesting dimensions to the story: words relating to animate concepts such as "nurse" or "cat" may have both masculine and feminine versions; the distributional environment of a word contains many more explicit gender cues; and inanimate concepts are also assigned gender. All of these factors interact in complicated ways. In this work we focus on purely grammatical gender—the gender that is assigned to inanimate nouns—and its effect on their resulting representations.

[2] in English word2vec embeddings (Mikolov et al., 2013) trained on Wikipedia.

[3] The gender direction is chosen to be the top principal component (PC) of ten gender pair difference vectors.

Grammatical Gender Bias in Word Embeddings Grammatical gender is manifested in a similar way to social bias. For example, when projected on the Italian gender direction $\overrightarrow{lui} - \overrightarrow{lei}$ (Italian equivalents of "he" and "she"), the word "secolo" (*century*, masculine) has positive bias of 0.073, while the word "zuppa" (*soup*, feminine) has negative bias of -0.079.[4]

We attribute this behavior to grammatical agreement. Since the context of different-gender nouns is expected to be very different because of the agreement of the surrounding words, and since the resulting representations are based on the context of the word, we expect grammatical gender to play a role in the representations—nouns with the same gender are expected to be closer together than nouns with different gender. For inanimate nouns, this behavior is undesired.

Word Embeddings and Morphology Word embeddings were shown to capture grammatical and morphological properties. Avraham and Goldberg (2017) show that standard training of word embeddings in Hebrew captures also morphological properties and that using the lemmas when composing the representations helps to better capture semantic similarities. Similarly, Basirat and Tang (2018) show that typical grammatical features are captured by Swedish word embeddings.

Cotterell et al. (2016) treat the sparsity problem of morphologically rich languages in word embedding. They present a Gaussian graphical model to smooth representations of observed words and extrapolate representations for unseen words using morphological resources. With similar motivation, Vulić et al. (2017) use morphological constraints in English in order to pull inflectional forms of the same word closer together and push derivational antonyms farther apart. Finally, Salama et al. (2018) enhance Arabic word embeddings by incorporating morphological annotations.

3 Grammatical Gender Affects Word Representations

As a first step, we aim to verify that the representation of inanimate nouns in gender-marking languages is indeed affected by their grammatical gender. Since English does not have grammatical gender, a natural approach would be to use it as a reference when measuring this phenomenon.

[4] in Italian word2vec embeddings (Mikolov et al., 2013) trained on Wikipedia.

3.1 Inanimate Noun Pairs from SimLex-999

We take the inanimate noun portion of the SimLex-999 dataset (Hill et al., 2015), a gold standard resource for evaluating distributional semantic models. This dataset has an English version, and also German and Italian versions (Leviant and Reichart, 2015), and includes both similar and dissimilar word pairs, with human-assigned similarity judgments for each pair. This gives us 529 pairs of English words, along with high quality translations to Italian and German. We manually associate the Italian and German words with their grammatical gender.

3.2 Differences in Similarities

We divide the pairs in the gender-marking (GM) language (be it German or Italian) into two sets: (1) pairs of nouns that have the same gender in the GM language; (2) pairs of nouns that have different gender in the GM language. The respective English pairs are split in the same way, according to the gender of the nouns in the GM language. Thus, we end up with two sets of pairs in a GM language and their translations to English. Note that the English sets are different when used as a reference for German and Italian, since the split depends on the gender in the respective language.

For each set we compute the average of the cosine similarity of all word pairs within it. If gender plays a role in the representation of words, and indeed brings same-gender words closer together while keeping different-gender words farther apart, we expect to see a significant difference between the average similarity of the set of same-gender nouns and the set of different-gender nouns. As mentioned above, we compute these averages for English as a reference, where we expect a low difference between the two sets. Table 1 shows the results for Italian and German, compared to English. Indeed, in both cases, the difference between the average of the two sets is much bigger.

3.3 Rank in Nearest Neighbor List

We take the same sets as before, and for each pair in them we compute the rank of the second word in the nearest neighbor list of the first word and vice versa. For example, for the pair "parola" (*word*) and "dizionario" (*dictionary*) in Italian, we compute the rank of "dizionario" in the list of nearest neighbor of "parola" and the rank of "parola" in

	Italian	En	German	En
Same Gender	0.442	0.424	0.491	0.446
Different Gender	0.385	0.415	0.415	0.403
difference	0.057	0.009	0.076	0.043

Table 1: Averages of similarities of pairs with same gender vs. different gender, along with the respective averages in English. The last row (difference) is the difference between the averages of the two sets.

the list of nearest neighbors of "dizionario".

We then compare the average ranking in each set, with English as the reference. If the gender affects the similarities between words, we expect same-gender pairs to have lower average than different-gender pairs (remember that the closest word is at the lowest rank: 1). Table 2 shows the results for Italian and German, compared to English. As expected, the average ranking of same-gender pairs is significantly lower than that of different-gender pairs, both for German and Italian, while the difference between the sets in English is much smaller.

4 Debiasing Methods do not Work

As mentioned above, grammatical gender bias shares some aspects with social gender bias. Keeping that in mind we first turn to use these existing methods of gender-debiasing in English word embeddings.

Bolukbasi's method (2016) requires sets of pairs that define the gender direction. For this we use their predefined pairs, since we target grammatical gender bias, which we have demonstrated to be similar to social gender bias. In addition, a predefined set of inherently-neutral words is also needed: these are the words that will be debiased by the algorithm. As a first step, and in order to estimate the feasibility of using this method for reducing the grammatical gender bias, we use the set of the inanimate nouns from SimLex-999 as our set of inherently-neutral words.[5]

The algorithm worked well in the sense that the bias of all inanimate nouns, when measured by their projection on the gender dimension, became zero. However, it also failed: the similarities between the inanimate nouns themselves hardly changed. Table 3 depicts the average similarities in Italian before and after debiasing.

This suggests that the information about the gender is deeply embedded in the representation and is not easy to remove in a post-processing phase. Specifically, zeroing the projection of a word's vector on the gender direction is not enough in order to remove all gender information from the word's representation. The fact that similarities between words hardly change implies that the projection on the gender direction is not the only indication of gender. These results align with the findings discussed in Gonen and Goldberg (2019).

We conclude that focusing on the projection of vectors on the gender direction is not the right way to go, and we opt to removing gender inflections from the context before training. We describe this in detail in the next section.

5 Removing Gender Inflection from the Context

As mentioned above, words in the surroundings of gender-marked nouns (e.g. articles, adjectives) are often inflected to agree with the gender of the noun they relate to. As we hypothesize that most of the effect shown in Section 3 is caused by this gender agreement, we try several schemes that aim to remove gender signals from the context.

A straight-forward approach would be to lemmatize all the words, which will remove all gender signals from the context of a word. However, this approach has two main downsides: 1) We would like to have a representation for all the words in the vocabulary, but changing also the target words will reduce the vocabulary size and result with missing words (we will no longer have different masculine and feminine forms for any word); 2) Lemmatization removes not only gender information, but also additional information (such as number and tense). While gender assignment is arguably arbitrary, and does not translate to an actual physical property of inanimate nouns in reality, other properties that agree with the noun, such as number, do hold in reality and signify actual properties of the target noun, which we prefer to preserve.

Thus, a better approach would be to neutralize gender signals from the context alone, keeping the target words intact. This way we do not change the resulting embedding vocabulary. This can be done using: 1) lemmatizing all the context words, where we lose additional information, as

[5] If this method doesn't mitigate the bias we showed in the previous section, then using inherently-neutral words extracted from the vocabulary automatically cannot possibly work as well.

	Italian			German		
	Same-gender	Diff-Gender	difference	Same-gender	Diff-Gender	difference
7–10	Og: 4884 Db: 5523 En: 6978	Og: 12947 Db: 7312 En: 2467	Og: 8063 Db: 1789 En: -4511	Og: 5925 Db: 7653 En: 4517	Og: 33604 Db: 26071 En: 8666	Og: 27679 Db: 18418 En: 4149
4–7	Og: 10954 Db: 12037 En: 15891	Og: 15838 Db: 12564 En: 17782	Og: 4884 Db: 527 En: 1891	Og: 19271 Db: 24845 En: 13282	Og: 27256 Db: 22970 En: 17649	Og: 7985 Db: -1875 En: 4367
0–4	Og: 23314 Db: 26386 En: 57278	Og: 35783 Db: 28067 En: 53053	Og: 12469 Db: 1681 En: -4225	Og: 50983 Db: 60603 En: 41509	Og: 85263 Db: 79081 En: 62929	Og: 34280 Db: 18478 En: 21420

Table 2: Averages of rankings of the words in same-gender pairs vs. different-gender pairs for Italian and German, along with their differences. **Og** stands for the original embeddings, **Db** for the debiased embeddings, and **En** for English. Each row presents the averages of pairs with the respective scores in SimLex-999 (0–4, 4–7, 7–10).

	Italian			
	Original	Debiased	English	Reduction
Same Gender	0.442	0.439	0.424	–
Different Gender	0.385	0.390	0.415	–
difference	0.057	0.049	0.009	**16.67%**

Table 3: Averages of similarities of pairs with same vs. different gender in Italian compared to the debiased version using Bolukbasi's (2016) method. The last row is the difference between the averages of the two sets. "Reduction" stands for gap reduction after debiasing.

discussed above; 2) changing all the context words to the same gender, while keeping all other features of the words intact. Once the whole context is of the same gender, we essentially lose the gender information altogether as all nouns have similar context, regardless their gender.[6]

5.1 The proposed approaches

We experiment with both lemmatization of context words and gender change of context words.

Lemmatization of Context Words When training word2vec (Mikolov et al., 2013), we use a morphological analyzer to identify the lemmas of words, and replace context words, but not target words, with their lemmas.

Gender Change of Context Words When training word2vec, we choose a gender (for example, masculine) and change all context words to that gender: each word that is identified as being of a different gender (in Italian: feminine, in German: feminine or neutral), is changed to its masculine form. This is also done using a morphological analyzer: when we identify a non-masculine analysis, we search for a masculine one that shares the same lemma and features.

In general, we found Italian to work better with gender change, and German to work better with lemmatization. We report full results in Section 6.

5.2 Challenges

While conceptually simple, fully neutralizing gender information is more challenging than it initially appears, and requires careful attention to "get right". We describe some cases in which gender information can leak.

Human Curator Choices The morphological analyzer sometimes assigns different lemmas to an opposite-gender pair, as a result of human curater design choices. For example, in Italian, "delle" is the feminine of "dei", but they are assigned the lemmas "della" and "del", respectively. Such cases leak gender signal in both cases of lemmatization and gender change: (1) When lemmatizing, each of the words gets a different lemma, manifesting the gender. (2) When changing the gender, the opposite-gender form of the word is not identified as these words do not share lemma, and the words stay unchanged.

This was very prominent in some high-frequency Italian words, and dealt with by fixing the analyzer: we identified all forms without a corresponding gendered-pair, manually aligned them, and assigned each pair a shared and unique lemma. This fix dramatically improved results when using either lemmatization or gender change.

Gender-Ambiguous Word Forms Many word forms have several morphological analyses, resulting in different lemmas. Inspecting this ambiguity reveals two specific issues, in German and in

[6]Context nouns are also kept unchanged since nouns do not agree with other nouns in their context, both in Italian and in German. Notably, in German, we lose the noun-ness information when we lowercase the corpus (as all nouns in German begin with an uppercase letter).

Italian. First, many German words are ambiguous with respect to gender. For example, "eine" has a frequent feminine reading, but also a rare masculine one. When changing words to masculine, this word is identified as potentially masculine, and kept intact. The presence of the context word "eine" now leaks a feminine signal.[7]

Second, Italian has many cases of two words with a similar set of possible lemmas but with different gender. For example, "usato" and "usata" are masculine and feminine, respectively, and both have "usare" and "usato" as possible lemmas. If we select a consistent lemma for each word type, and end up selecting a different lemma for each of "usato" and "usata", we again leak signal regarding the original gender.

One solution would be to use context-sensitive lemmatization, that chooses the correct analysis in context. However, doing this accurately is still an open problem. Our proposed solution is to randomly sample a lemma per word token. This improved lemmatization results in Italian by 25%.

Multiple Opposite-gender Forms for a Word In some cases, a single word might have multiple forms in the opposite gender. For example, the Italian "delle" is the feminine form of both "dei" and "degli", depending on the phonetic context. In this case, the former is much more common than the latter. A naive approach that chooses to convert "delle" to "degli" essentially keeps the feminine signal for these cases: every instance of "delle" changes to "delgli", which marks masculine nouns in much less common cases, while most masculine nouns are usually accompanied with the more common word "dei".

Ideally, when changing the gender of a word, we want to change a word by another word with a similar frequency, otherwise, the gender signal will be manifested in the frequency mismatch, as in the example above.

We deal with this issue using the following heuristic: when changing to masculine form (or any other gender form), for each word we first find all its possible masculine forms. Then, we check the frequency of the original word in the corpus, and choose the option with the closest frequency to it. This indeed yields better results: when not addressing the frequency issue in Italian, we are able to reduce the effect only by 35.42% (compared to 91.67%, see Section 6 for more details).

6 Results

We experimented with different schemes for each language, measuring their success at removing gender bias of inanimate nouns with respect to English.[8]

For German, we found lemmatization to work better than gender change. In Italian gender change got better results. Specifically, changing to feminine got much better results than changing to masculine, probably due to less ambiguity when changing to feminine in some very common articles (see full manual mapping in the Appendix), resulting in fewer cases in which the gender signal leaks through the frequencies of the changed words, as explained above. In addition, the manual fixes to the lemmatizer were crucial to get satisfying results for both methods.

While some of these findings depend on the specific morphological analyzer in use, the challenges and issues we demonstrate are relevant in any case.

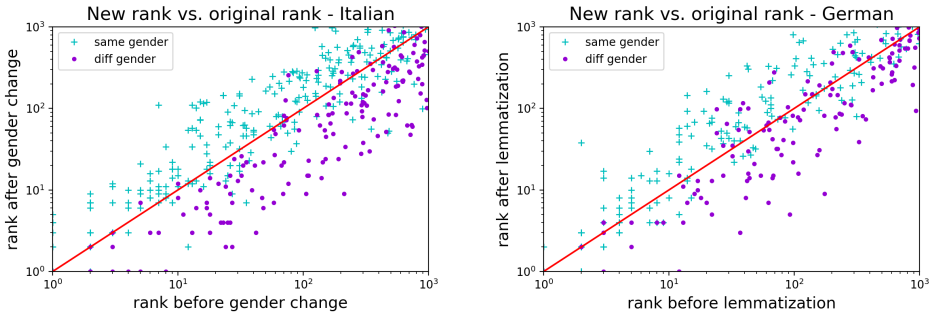
6.1 Reduction in Gender Bias

Differences in Similarities We repeat the experiment in Section 3.2—computing the average of pair similarities in each of the sets defined in Section 3, this time with the embeddings trained after removing gender signal from the context (debiasing). Table 4 shows the results for Italian and German, compared to English, both for the original and the debiased embeddings (for each language we show the results of the best performing debiased embeddings). As expected, in both languages, the difference between the average of the two sets with the debiased embeddings is much lower. In Italian, we get a reduction of 91.67% of the gap with respect to English. In German, we get a reduction of 100%. Note that for both languages, the main change is in the set of different-gender pairs, and not in the same-gender pairs. This makes sense as same-gender words have similar contexts both before and after our intervention, but different-gender words have different contexts before, but much more similar contexts after.

For comparison, in Italian we got 12.50% reduction when using the lemmatization scheme,

[7] A possible solution would be to replace words with their lemmas whenever we identify both feminine and masculine analyses. This did not improve results in practice.

[8] We used state-of-the-art morphological analyzers for both languages. Full implementation details can be found in the appendix.



(a) (b)

Figure 1: The new rank of a word in the nearest neighbor list of its paired word. In cyan (+) – pairs with the same gender, in purple (·) – pairs with different gender. Most words with same-gender are located above $y = x$ (were drifted apart), while most words with different-gender are located below it (got closer together).

	Italian				German			
	Original	Debiased	English	Reduction	Original	Debiased	English	Reduction
Same Gender	0.442	0.434	0.424	–	0.491	0.478	0.446	–
Different Gender	0.385	0.421	0.415	–	0.415	0.435	0.403	–
difference	0.057	0.013	0.009	**91.67%**	0.076	0.043	0.043	**100%**

Table 4: Averages of similarities of pairs with same vs. different gender in Italian and German compared to English. The last row is the difference between the averages of the two sets. "Reduction" stands for gap reduction when removing gender signals from the context.

and 68.75% reduction when lemmatizing with the addition of the manual mapping. For German, the best result using gender change was a reduction of 48.48%, achieved by changing to neutral.

Rank in Nearest Neighbor List We repeat the experiment shown in Section 3.3—for each pair we compute the rank of the second word in the nearest neighbor list of the first word and vice versa. Then we compare the average ranking in each of the defined sets. Table 2 shows the results for Italian and German, both for the original and the debiased embeddings. As we expect, the difference between the average ranking of the two sets drops significantly for both languages.

In order to get a better picture of how the rankings of the different words change as a result of the gender signal removal, we take all pairs (and the inverted pairs). For each pair we plot the new rank of the second word in the nearest neighbors list of the first word as a function of its original rank before debiasing. Points above $y = x$ are of words that got a higher rank (lower in the list, farther from the first word), while points below this line are of words that got a lower rank (higher in the list, closer to the first word). Figure 1 shows these plots for Italian and German. As expected, most words of same-gender pairs are located above the line (were drifted apart), while most words of different-gender pairs are located below the line (got closer together).

6.2 Improvement in Word Similarities

Qualitative Evaluation As a qualitative evaluation, we take several words for SimLex-999 and look at their top-10 nearest neighbor lists, before and after applying our method. In Table 5 we show the top-10 lists for the words *vaso* ("*jar*"-masculine) in Italian, and *welt* ("*world*"-feminine) in German. It is evident that the words that are added to the list, are better correlated with the target word than those that are removed. Two additional words appear in the Appendix.

Evaluation on Simlex and WordSim-353 We evaluate the quality of the grammatical-gender-neutralized embeddings using two datasets for each language: SimLex-999 (Hill et al., 2015; Leviant and Reichart, 2015) and WordSim-353 (Finkelstein et al., 2002; Leviant and Reichart, 2015). Table 6 shows the results for Italian and German for both datasets, compared to the original embeddings. In both cases, the new embeddings perform better than the original ones.

Cross-lingual Word Embeddings Studies in language and cognition suggest that humans share

Italian		German	
vaso (jar-masculine)		welt (world-feminine)	
Orig	Debias	Orig	Debias
coccio	vasi	welt”	europas
recipiente	ciotola	europas	welt”
otre	bacinella (basin)	scheibenwelt	scheibenwelt
cinerario	recipiente	hässlichsten	universum (universe)
vasetto	coccio	*erde* (earth)	menschheitsgeschichte (human history)
bacile (basin)	cinerario	*weltgeschichte* (world history)	hässlichsten
kantharos	otre	*klügste* (wisest)	menschheit (mankind)
vasi	vasetto	*klügsten* (wisest)	schwarzafrikas
vassoio (tray)	brocca (pitcher)	schwarzafrikas	parallelwelten (parallel worlds)
coperchio (cover)	scodella (bowl)	*lustigsten* (funniest)	ulldart

Table 5: Examples of top-10 nearest neighbor lists for words in Italian and in German, before and after debiasing. In red (italic) are words that were removed from the list, and in blue (underlined) are words that were added to it. Translations to English (Google Translate) for the changed words are in parenthesis, when different from source.

	Italian		German	
	Orig	Debias	Orig	Debias
SimLex	0.280	**0.288**	0.343	**0.356**
WordSim	0.548	**0.577**	0.547	**0.553**

Table 6: Results on SimLex-999 and WordSim-353, in Italian and German, before and after debiasing.

	Italian		German	
	→ En	En →	→ En	En →
Orig	58.73	59.68	47.58	50.48
Debias	**60.03**	**60.96**	**47.89**	**51.76**

Table 7: Cross-lingual embedding alignment in Italian and in German, before and after debiasing.

a common semantic space, regardless of their native language (Youn et al., 2016). To the extent that embeddings capture the semantics of words, we can thus expect embedding spaces to have a similar structure across languages. Youn's statement concerns concepts and not words, however, and concepts can surface in many different forms in language, which interferes with how well embedding spaces align across languages (Søgaard et al., 2018). Thus, we expect grammatical gender to have a negative impact on alignability.

We explore this matter through the task of cross-lingual embedding alignment, wherein a cross-lingual embedding space is learned through an alignment of independently pre-trained monolingual embeddings for a directed pair of languages. The quality of cross-lingual embeddings learned this way can be evaluated intrinsically on the task of bilingual dictionary induction (BDI). BDI queries the cross-lingual embedding space with a seed of words in one language, retrieves their counterparts among the words in the other language[9] and evaluates the precision of the produced translations against a set of gold standard targets. We carry out experiments using the supervised variant of the MUSE embedding alignment system (Conneau et al., 2018) and report results on the inanimate portion of SimLex-999. We train a cross-lingual embedding alignment between English and either German or Italian, using the original and the debiased embeddings for these two languages. The results reported in Table 7 show that precision on BDI indeed increases as a result of the reduced effect of grammatical gender on the embeddings for German and Italian, i.e. that the embeddings spaces can be aligned better with the debiased embeddings.

7 Conclusion

We show that grammatical gender impacts word embeddings of inanimate nouns, both in Italian and in German, causing the similarities between words to change according to having same or different gender: the representations of same-gender words are closer together than representations of different-gender words.

We show that this effect can be almost completely removed when neutralizing gender signals in the context during training of the word embeddings. While most works in our field nowadays try to be language-independent, this is not always the right way to go: successfully removing those gender signals is not trivial to do and a language-specific morphological analyzer, to-

[9]This is done by minimizing a distance metric, most commonly, CSLS (Conneau et al., 2018).

gether with careful usage of it, are essential for achieving good results.[10]

In addition, this work serves as a reminder that languages other than English have different properties that are rarely dealt with when processing English. These aspects should be taken into account when dealing with morphologically reach languages, as not all models and algorithms for English can transfer directly to other languages.

8 Acknowledgements

The work was supported by the Israeli Science Foundation (grant number 1555/15) and by and the European Research Council (ERC Starting Grant iExtract 802774). We thank Valentina Pyatkin for the help with the Italian manual mapping.

References

Oded Avraham and Yoav Goldberg. 2017. The interplay of semantics and morphology in word embeddings. In *Proceedings of EACL*.

Ali Basirat and Marc Tang. 2018. Lexical and morpho-syntactic features in word embeddings-a case study of nouns in swedish. In *ICAART*.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*.

Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word Translation Without Parallel Data. In *Proceedings of ICLR 2018*.

Greville G Corbett. 1991. Gender.

Greville G Corbett. 2006. *Agreement*. Cambridge University Press.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of ACL*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Transactions on information systems*.

Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *Proceedings of NAACL-HLT*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3).

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4).

Ira Leviant and Roi Reichart. 2015. Separated by an un-common language: Towards judgment language informed vector space modeling. *arXiv:1508.00106*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv:1301.3781*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*.

Rana Aref Salama, Abdou Youssef, and Aly Fahmya. 2018. Morphological word embedding for arabic. In *ACLing*.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of ACL*.

Ivan Vulić, Nikola Mrkšić, Roi Reichart, Diarmuid Ó Séaghdha, Steve Young, and Anna Korhonen. 2017. Morph-fitting: Fine-tuning word vector spaces with simple language-specific rules. *arXiv:1706.00377*.

Hyejin Youn, Logan Sutton, Eric Smith, Cristopher Moore, Jon F. Wilkins, Ian Maddieson, William Croft, and Tanmoy Bhattacharya. 2016. On the universal structure of human lexical semantics. In *NIPS*.

Nasser Zalmout and Nizar Habash. 2017. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for arabic. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. 2018. Learning gender-neutral word embeddings. *arXiv preprint arXiv:1809.01496*.

Ran Zmigrod, Sebastian J Mielke, Hanna Wallach, and Ryan Cotterell. 2019. Counterfactual data augmentation for mitigating gender stereotypes in languages with rich morphology. *arXiv preprint arXiv:1906.04571*.

[10] Indeed, before implementing the specific fixes described in Section 5, the reduction compared to English when naively changing to masculine was substantially smaller, 35.42% reduction compared to 91.67% in Italian, and 12.12% compared to 100.00% (with lemmatization) in German.

Active Learning via Membership Query Synthesis for Semi-supervised Sentence Classification

Raphael Schumann
Institute for Computational Linguistics
Heidelberg University, Germany
rschuman@cl.uni-heidelberg.de

Ines Rehbein
Leibniz ScienceCampus
Heidelberg/Mannheim
rehbein@ids-mannheim.de

Abstract

Active learning (AL) is a technique for reducing manual annotation effort during the annotation of training data for machine learning classifiers. For NLP tasks, pool-based and stream-based sampling techniques have been used to select new instances for AL while generating new, artificial instances via Membership Query Synthesis was, up to know, considered to be infeasible for NLP problems. We present the first successful attempt to use Membership Query Synthesis for generating AL queries for natural language processing, using Variational Autoencoders for query generation. We evaluate our approach in a text classification task and demonstrate that query synthesis shows competitive performance to pool-based AL strategies while substantially reducing annotation time.

1 Introduction

Active learning (AL) has the potential to substantially reduce the amount of labeled instances needed to reach a certain classifier performance in supervised machine learning. It works by selecting new instances that are highly informative for the classifier, so that comparable classification accuracies can be obtained on a much smaller training set. AL strategies can be categorized into pool-based sampling, stream-based sampling and Membership Query Synthesis (MQS). The first two strategies sample new instances either from a data pool or from a stream of data. The third, MQS, generates artificial AL instances from the region of uncertainty of the classifier. While it is known that MQS can reduce the predictive error rate more quickly than pool-based sampling (Ling and Du, 2008), so far it has not been used for NLP tasks because artificially created textual instances are uninterpretable for human annotators.

We provide proof of concept that generating highly informative artificial training instances for text classification is feasible. We use Variational Autoencoders (VAE) (Kingma and Welling, 2013) to learn representations from unlabeled text in an unsupervised fashion by encoding individual sentences as low-dimensional vectors in latent space. In addition to mapping input sequences into latent space, the VAE can also learn to generate new instances from this space. We utilize these abilities to generate new examples for active learning from a region in latent space where the classifier is most uncertain, and hand them over to the annotator who then provides labels for the newly created instances.

We test our approach in a text classification setup with a real human annotator in the loop. Our experiments show that query synthesis for NLP is not only feasible but can outperform other AL strategies in a sentiment classification task with respect to annotation time.

The paper is structured as follows. We first review related work (§2) and introduce a formal description of the problem (§3). Then we describe our approach (§4), present the experiments (§5) and analyze the results (§6). We discuss limitations and possible further experiments (§7) and finally conclude our findings (§8).

2 Related work

Membership query synthesis was introduced by Angluin (1988) and describes a setting where the model generates new queries instead of selecting existing ones. Early experiments in image processing (Lang and Baum, 1992), however, showed that the generated queries are hard to interpret by human annotators. This holds true even for recent approaches using Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) to create uncertain instances (Zhu and Bento, 2017; Huijser and van Gemert, 2017). In contrast to im-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 472–481
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

age processing, discrete domains like natural language do not exhibit a direct mapping from feature to instance space. Strategies that circumvent this problem include the search for nearest (observed) neighbors in feature space (Wang et al., 2015) or crafting queries by switching words (Awasthi and Kanade, 2012).

Sentence representation learning (Kiros et al., 2015; Conneau et al., 2017; Subramanian et al., 2018; Wang et al., 2019) in combination with new methods for semi-supervised learning (Kingma et al., 2014; Hu et al., 2017; Xu et al., 2017; Odena, 2016; Radford et al., 2017) have shown to improve classification tasks by leveraging unlabeled text. Methods based on deep generative models like GANs or VAEs are able to generate sentences from any point in representation space. Mehrjou et al. (2018) use VAEs to learn structural information from unlabeled data and use it as an additional criterion in conventional active learning to make it more robust against outliers and noise.

We use VAEs to generate AL queries from specific regions in latent space. To ensure that the generated instances are not only informative for the ML classifier but also meaningful for the human annotator, we adapt the approach of Wang et al. (2015) (see §3.1). In contrast to their work, however, we do not *sample* existing instances from the pool that are similar to the synthetic ones but directly *generate* the new queries. To our best knowledge, our work is the first to present positive results for Membership Query Synthesis for text classification.

3 Background

3.1 Query Synthesis and Nearest Neighbors

Arbitrary points in feature space are hard to interpret for humans. To evade this problem, Wang et al. (2015) use the nearest neighbor in a pool of unlabeled data as a representative which is then presented to the human annotator. To identify uncertain points along the separating hyperplane of an SVM the following approach is proposed. First the location of the decision boundary is approximated by a binary-search like procedure. An initial *Opposite Pair* (z_+, z_-) is formed by centroid c_+ and centroid c_- of positive and negative labeled instances respectively. The mid point z_s is queried and, depending on the annotated label l, replaces the corresponding z_l. This step is repeated b times, reducing the distance between the initial centroids by a factor of 2^b. Figure 1a depicts this process. Then the mid-perpendicular vector of the *Opposite Pair* is calculated by using the Gram-Schmidt process to orthogonalize a random vector z_r and normalize its magnitude to λ. The new point $z_s = z_r + (z_+ + z_-)/2$ is close to the decision boundary and queried for its class. Depending on the receive label the point z_s replaces z_+ or z_- in the *Opposite Pair*. This process (Figure 1b) is repeated until $n - b$ points along the separating hyperplane are queried.

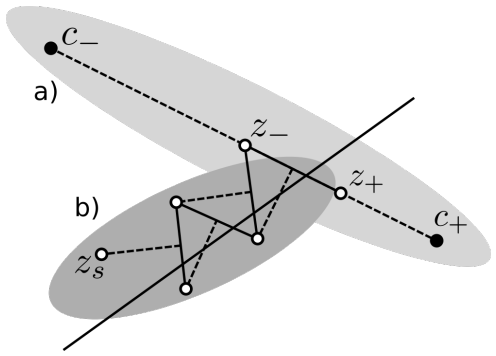


Figure 1: **a)** finds *Opposite Pair* close to the decision boundary. **b)** identify points close to the decision boundary.

3.2 VAE for Sentence Generation

The Variational Autoencoder is a generative model first introduced by Kingma and Welling (2013). Like other autoencoders, VAEs learn a mapping $q_\theta(z|x)$ from high dimensional input x to a low dimensional latent variable z. Instead of doing this in a deterministic way, the encoder learns the parameters of e.g. a normal distribution. The desired effect is that each area in the latent space has a semantic meaning and thus samples from $p(z)$ can be decoded in a meaningful way. The decoder $p_\theta(x|z)$, also referred to as $dec(z)$, is trained to reconstruct the input x based on the latent variable z. In order to approximate θ via gradient descent the reparametrization trick (Kingma and Welling, 2013) was introduced. This trick allows the gradient to flow through non-deterministic z by separating the discrete sampling operation. Let μ and σ be deterministic outputs of the encoder $q_\theta(\mu, \sigma|x)$:

$$z = \mu + \sigma \odot \epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, I) \qquad (1)$$

and $\odot$ is the element-wise product. To prevent the model from pushing σ close to 0 and thus falling back to a deterministic autoencoder, the objective is extended by the Kullback-Leibler (KL) diver-

gence between prior $p(z)$ and $q(z|x)$:

$$\mathcal{L}(\theta; x) = -KL(q_\theta(z|x)||p(z)) + \mathbb{E}_{q_\theta(z|x)}[logp_\theta(x|z)]. \quad (2)$$

Bowman et al. (2016) apply this idea for sentence generation using an RNN as encoder and decoder. They observe that a strong auto-regressive language modeling ability in the decoder reduces the information stored in the latent variable, right up to a complete collapse of the KL term. They explore different techniques to weaken the decoder, like word dropout or KL term weight annealing, as possible solutions. This guarantees a semantically rich latent variable and good sentence generation ability. Below, we describe how to combine both techniques in order to generate meaningful queries for Membership Query Synthesis.

4 Active Learning Schedule

We train a Variational Autoencoder on an unlabeled corpus of sentences. The text classification task is performed on a binary sentiment dataset split into training, development and test set. As depicted in Figure 2, the sentences in the classification dataset are vectorized using the VAE encoder which generates the latent variable z for each sentence x. This is done deterministically by dropping the σ term in Equation 1, further referred to as $z = enc(x)$.

Next, a *Learner* is trained to fit a linear hyperplane to separate the positive from the negative instances. We use the procedure described in §3.1 to select new query points for AL. But instead of searching for the nearest neighbor in the pool, we decode the point $x = dec(z)$ into a human readable sentence which is then handed over to the human annotator. The annotator assigns a binary label to the instance and the next query point is calculated.

One important parameter for active learning determines how many new instances are to be selected in each AL iteration. Wang et al. (2015) use a predefined number of instances to be selected along the hyperplane. Because we know that a Gaussian prior is imposed on the feature space, we instead stop the selection process when the magnitude of z_s exceeds the expectation. The expected distance of a point sampled from the k-dimensional Gaussian prior to the origin is $\sqrt{\mathbb{E}[\chi^2_k]} = \sqrt{k}$. Then the schedule restarts, learning a new decision boundary, and ultimately terminates when the annotation budget is exhausted. We refer to this method as *gen_wang*. When nearest neighbor search is used instead, we refer to the selection method as *nn_wang*.

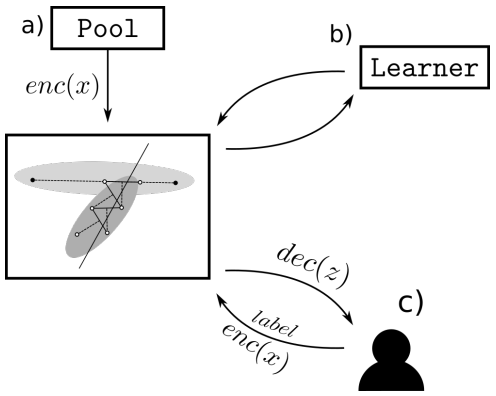


Figure 2: **a)** Instances in corpus are encoded to latent space. **b)** Learner fits a hyperplane to separate points. **c)** Query points selected by method described in Fig.1. Decoder translates point in latent space to human readable sequence. Annotator chooses label for instance.

In addition, we explore a method, *gen_uniform*, where step b) in Figure 1 is reduced to generating only one midperpendicular vector with a magnitude drawn from a uniform distribution. In each iteration this vector will point to a random direction with a different magnitude, selecting diverse points close to the hyperplane. The maximum magnitude is set in a way that the resulting point is not further away than $\sqrt{k}$ from the origin. Similar to above we refer to this method as *nn_uniform* when using nearest neighbor search. The number of possible directions along the hyperplane grows with the size of the latent variable. With this modification we expect to explore more diverse points than following the same direction for several steps.

5 Experiments

In this section we want to explore how the ability to generate human readable sentences from arbitrary points in the feature space affects active learning performance. We compare our approach to a number of baselines (§5.3), where in each experiment we select/generate 500 instances, present them to a human annotator to get a label and evaluate the performance of each setting in a sentiment classification task. We start the active learning process with two utterances in the seed set, namely '*good movie*' and '*bad movie*'. The classifier is trained to separate instances with positive sentiment from negative ones. The human anno-

Parameter	Value
vocabulary size	20.000
RNN cell size	512
embedding size	512
latent variable size	50
dropout	0.3
dropword	0.5
learning rate	0.005
epochs	20

Table 1: Training parameters for the Variational Autoencoder.

tator can skip neutral or uninterpretable instances. These skip actions also count towards the annotation budget.

5.1 Data

The data used in our experiments comes from two sources, (i) the SST2 (Socher et al., 2013) and (ii) SAR14 (Nguyen et al., 2014). We limit sentence length to a maximum of 15 words. This is motivated by lower training times and the tendency of vanilla VAEs not to perform well on longer sentences (Shen et al., 2019).

Sentiment task SST2 (Socher et al., 2013) is a binary sentiment classification dataset compiled from *rottentomatoes.com*. As we only consider sentences with up to 15 words, the sizes of the training, development and test sets are 3103, 380 and 814 instances, respectively.

Sentence pool The active learning pool consists of 1.2M unique sentences from the SAR14 dataset (Nguyen et al., 2014). SAR14 contains 234k movie reviews from IMDB. The data is annotated on review level, which prevents us from removing single neutral sentences. Although the datasets stem from different sources, there is a small overlap. These sentences are removed from the pool.

5.2 Training

Variational Autoencoder Table 1 lists the parameter used for the VAE. For training we limit the vocabulary of the VAE to the top 20k words. Encoder and decoder RNN have layer normalized (Ba et al., 2016) LSTM cells (Hochreiter and Schmidhuber, 1997) with size 512. As additional regularization we set weight dropout to 0.3 (Srivastava et al., 2014). Input embeddings are also of size 512, which allows us to share the embedding weights with the softmax weights of the output layer (Press and Wolf, 2016). To prevent posterior collapse we use logistic annealing of the KL term weight and weaken the decoder by applying word dropout with probability 0.5 (Bowman et al., 2016). The model is trained using the Adam optimizer (Kingma and Ba, 2014) with an initial learning rate of 0.005. Once the KL term weight is close to 1, the learning weight is linearly decreased to 0. The training stops after 20 epochs and the latent variable z has $k = 50$ dimensions. The trained VAE achieves a reconstruction loss of 45.3 and KL divergence of 13.2 on the SST2 training set.

Learner The *Learner* is an SVM[1] with linear kernel. Each instance is represented as the latent variable z learned by the autoencoder. The latent variable is a vector with 50 dimensions and the SVM is trained on this representation. We calculate classification performance on the reduced SST2 test set and report F1-scores.

Generator The generator is the decoder of the VAE described above. Once a point z in feature space is selected, it is used as the input of the decoder $x = dec(z)$ which generates the human readable sentence x in an autoregressive way.

5.3 Baselines

We compare our approach to Membership Query Synthesis for text classification to four baselines. The first baseline selects instances from the pool by *random* choice. The *least confidence* baseline computes the distance of the instances in the pool to the separating hyperplane and chooses the one closest to the hyperplane. The third and fourth baseline follow the procedure described in §4 but search for the nearest neighbor (*nn_uniform*, *nn_wang*) instead of synthesising the exact query point. Nearest neighbor is defined by the minimal euclidean distance between the query point and the latent representation of the pool instance.

5.4 Annotation

The instances selected or generated by any model or baseline are annotated manually by one human coder.[2] Although the pool data has labels on the review level, we do not use these labels in our experiments. Positive reviews can include negative

[1] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[2] The first author of this paper.

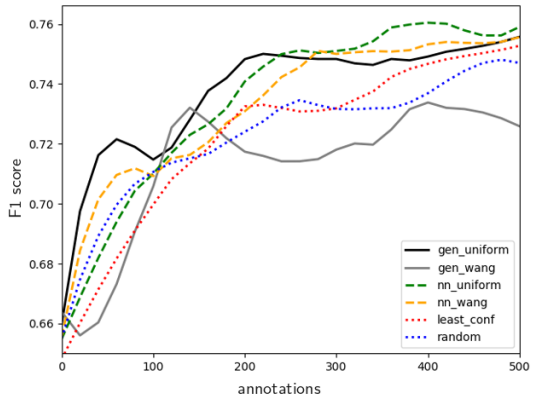


Figure 3: F1-Score as a function of annotation steps (including skipped queries). Averaged over 3 runs.

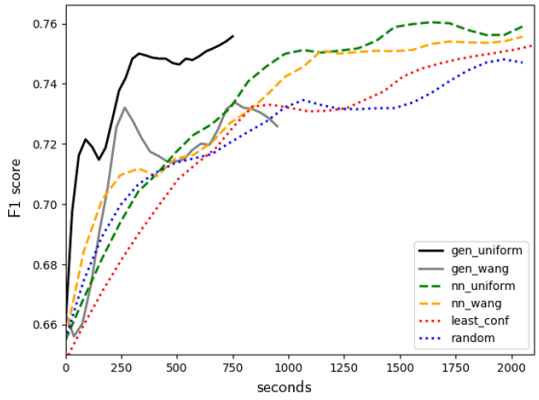


Figure 4: F1-scores as a function of annotation time. Results averaged over 3 runs.

sentences and vice versa. This means that using document-level labels would introduce noise and might impair the baselines. During each of the three experimental runs, all models and baselines are annotated simultaneously by the same person. The annotator is presented with one instance at a time and has no information which of the models has produced each particular instance. Once a label is selected, it is transmitted to the corresponding model and triggers the selection/generation of the next instance. Thus, at any given time there is one unlabeled instance for each model or baseline. From this set of unlabeled instances, one instance is chosen randomly and presented to the annotator. This procedure is repeated until 500 instances are labeled for each model or baseline. Hiding the instance source from the annotator is intended to prevent any bias during the annotation process.

6 Results and Analysis

6.1 Classification Performance

F-scores as a function of annotated instances Figure 3 shows learning curves for the different AL strategies and baselines as a function of the number of annotation instances added to the training data. The *random* and *least_conf* baselines perform reasonably well. *Least_conf* struggles in the beginning, likely attributed to the minimal seed set. Once enough instances are labeled it catches up. *Gen_uniform* has a strong start but, after around 200 instances, is outperformed by the *nearest neighbor* approaches which yield the highest F1-scores. Among the nearest neighbor approaches, the *uniform* schedule ranks better than *wang*. The same behaviour is observed for the *generation* methods, although *gen_wang* produces the worst results overall. Overall, *gen_uniform* is competitive with respect to F1-scores and shows that sentences generated from points in the feature space are informative and useful for training a text classifier.

F-scores as a function of annotation time AL simulations have often been criticized for reporting unrealistic results, based merely on the number of annotated instances (see, e.g., Settles (2009), pp. 37 ff.). It is well known, however, that the number of annotated instances is often not a good predictor for the real annotation costs. AL strategies tend to select the *hard nuts* for human annotators and it is not unreasonable to assume that the annotation of N instances in an AL setup might take longer and thus might be more expensive than annotating the same number of *randomly* selected instances. Therefore, we also show learning curves as a function of annotation time (Figure 4).

The results show a clear advantage for the *generation* models. The reduction in annotation time is due to shorter query length and less neutral or noisy instances, as shown in Table 2. This speeds up the annotation by a significant margin while providing the *Learner* with informative instances, despite their short length.

Figure 5 shows that the length of generated instances increase over time and further exploration also hints that the generated length is correlated with the length of the sentences in the seed set.

As listed in Table 2, the *random* baseline reveals that 36.8 percent of sentences in the pool are neutral/artifacts and positive sentences outweigh negative ones by a factor of 2.6. This means that random sampling results in unbalanced datasets with far more positive examples. Our generation

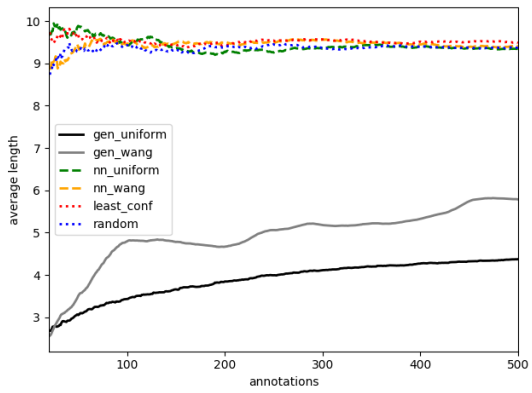


Figure 5: Development of average length of selected/generated instances as more instances are annotated.

method does not show this disadvantage. In contrast, the generated instances maintain a more balanced distribution of class labels and are less likely to be skipped. These are indicators that the selected points are close to the hyperplane and the VAE is able to generate coherent and highly informative sentences from them.

6.2 Computational Complexity

To assure a seamless annotation procedure, the supply of new instances has to be reasonably fast. The generation and selection of the next instance is dependant on the label of the previous instance. Because of this, there is no way to pre-fetch the next instance in the background and the annotator has to wait for the selection/generation process to finish before the next instance is presented for annotation. However, the runtime for pool-based AL methods is increasing with the pool's size. In contrast, the generation method presented in this work does not have this limitation.

The *least confidence* baseline has a complexity of $\mathcal{O}(n)$ where n is the number of instances in the pool. The complexity of *nearest neighbor* search without any approximation techniques like preclustering is also $\mathcal{O}(n)$. Query generation from an exact point with the decoder has a complexity of $\mathcal{O}(m)$ where m is the length of the sentence and $n >> m$. Because sentences have a natural length limit and in this work are capped to 15 words, one could argue that the complexity is $\mathcal{O}(1)$.

6.3 Generated Instances

Table 3 shows examples of generated instances using the *gen_uniform* method. Example 1-6 show prototypical positive and negative instances. Example 7 is ambiguous, caused by the decoder generating an unknown (UNK) token at the position where one would normally expect an evaluative adjective. We see this as an indicator that the point is positioned close to the hyperplane and thus the sentiment of the latent variable is ambiguous. We also observe instances with UNK token which still express a sentiment, as seen in Example 8 an 9. This can be interpreted as a placeholder for a named entity or, in other cases, a specifier like movie genre and does not impact the annotation process.

To explore the ability of the model to generate unseen instances we calculate the percentage of instances not seen in the pool. We only look at instances with an annotated sentiment label, because skipped examples often include noise and thus are unlikely to be present in the pool. 41 and 51 percent of labeled instances are newly generated by *gen_uniform* and *gen_wang* respectively. This provides more evidence that the model is capable of generating new and informative instances.

	% skips	M sec	M len	p/n
gen_uniform	28.1	1.4	4	1.7
gen_wang	20.9	1.9	5	1.2
nn_uniform	34.2	4.1	9	1.9
nn_wang	35.8	4.1	10	2.4
least_conf	39.0	4.2	10	2.1
random	36.8	4.1	9	2.6

Table 2: Percentage of skips (neutral or noisy sentences); Median annotation time in seconds; Median number of words in query; Ratio of positive to negative labels.

No.	Instance	Label
1.	the acting is excellent	1
2.	powerful and moving	1
3.	this movie is very enjoyable	1
4.	a complete mess	0
5.	nothing spectacular	0
6.	absolutely terrible !	0
7.	the plot is UNK	skip
8.	well done by UNK	1
9.	the UNK is a disappointment	0

Table 3: Example instances generated by *gen_uniform*. Label 1 for positive and 0 for negative class.

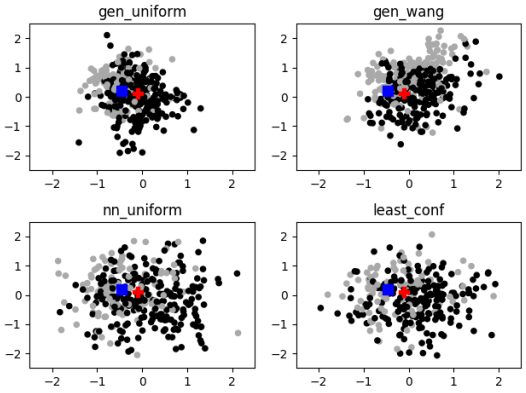


Figure 6: Plot of the 2 *most important* dimensions of selected/generated instances in latent space. Gray points indicate negative, black points positive labels. The blue square denotes '*bad movie*' and the red cross '*good movie*'.

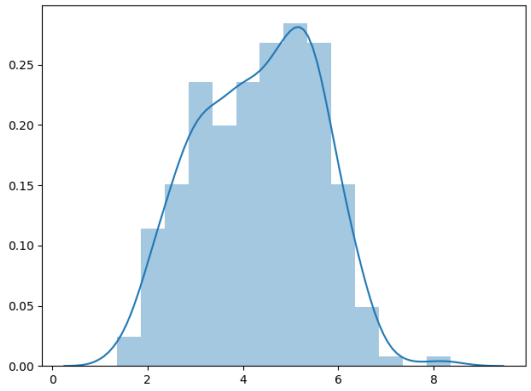


Figure 7: Distribution of euclidean distances between z before and $\hat{z}$ after re-encoding during *gen_uniform*.

6.4 Latent Space

To further analyze the behavior of the different AL strategies, we apply dimensionality reduction and visualize the instances in latent space (Figure 6).

The two largest absolute coefficients of the trained SVM's linear kernel identify the *most important* dimensions. Figure 6 plots the points, represented by theses two dimensions, selected by different active learning schedules. The generated instances lie densely around the seed points, while pool instances are more distributed. In *gen_wang* one can see how the instances are loosely following one direction similar to Figure1.

As indicated in Figure 2 a pool instance is represented as $z = enc(x)$. The same is true for the instances in the development, test and seed set. For the generated instances there are two options. If z is a point selected in feature space and $x = dec(z)$ is the decoded query sequence, the annotated instance can either be represented as z or as $\hat{z} = enc(x)$. In a perfect VAE z and $\hat{z}$ should be nearly identical. In practice however $\hat{z}$ ends up at a different location in feature space. Figure 7 depicts the distribution of distances between z and $\hat{z}$ generated with the *gen_uniform* method. We observe that models trained on $\hat{z}$ perform better than those trained on z, presumably because the test instances are represented the same way. To evaluate if $\hat{z}$ is still an informative point and not just positioned randomly in feature space, we train a model on actual randomly sampled points. The sampled point $z \sim \mathcal{N}(0, I)$ is decoded to query sequence x, labeled and subsequently re-encoded to $\hat{z} = enc(x)$. With the same amount of instances, this model performs much worse than *gen_uniform*, indicating that point $\hat{z}$ still preserves some of the informativeness of z. We thus assume that the closer $\hat{z}$ is to selected point z, the better the generation based active learning schedules will work.

7 Discussion

Related work in the context of semi-supervised learning has focused on developing methods to generate synthetic training instances for different tasks (Sennrich et al., 2016; Hayashi et al., 2018; Alberti et al., 2019; Winata et al., 2019), in order to accelerate the learning process. Sennrich et al. (2016) create artificial training instances for machine translation, using monolingual data paired with automatic back-translations. Their work obtains substantial improvements for several languages and has triggered many follow-up studies that apply the idea of back-translation to different tasks.

For example, Hayashi et al. (2018) augment the training data for attention-based end-to-end automatic speech recognition with synthetic instances, and Winata et al. (2019) generate artificial training examples to improve automatic speech recognition on code-switching material. Alberti et al. (2019) use a large number of synthetic instances to pre-train a Question Answering (QA) model that is then fine-tuned on the target QA dataset. Their approach results in significant improvements over models that are trained without the synthetic datapoints.

While these studies show that huge amounts of synthetic training data can crucially improve the

learning process, our approach uses a different paradigm. Instead of generating millions of synthetic data points, our method is data-lean and only needs a few hundred instances to improve the classifier. Another difference is that we do not rely on automatically generated labels but use human annotations instead. Due to the practical constraints of the active learning process, we need to keep the training time short enough so that the human annotator does not have to wait for the next set of instances to annotate. This rules out the use of computation-intensive models and large training sets. Given that we use an SVM for classification, we do not expect a strong effect for adding large numbers of additional training instances, given that the majority of those data points will not be positioned close to the decision boundary.

One of the main drawbacks of our work is its limitation to binary sentence classification. However, multi-class classification in an one-vs-rest schema is compatible with our method and worth further exploration. Another interesting direction for future work is the synthesis of data for more complex tasks like Natural Language Inference (NLI) or QA. This, however, requires modifications to the structure of the autoencoder and exceeds the scope of this work.

Membership Query Synthesis might also be an interesting approach for tasks where the automatic extraction of large amounts of unlabelled data is not straight-forward. One example that comes to mind is the detection of offensive language or 'hate speech', where we have to deal with highly unbalanced training sets with only a small number of positive instances, and attempts to increase this number have been shown to result in systematically biased datasets (Davidson et al., 2019; Wiegand et al., 2019). Table 2 suggests that the generator produces instances with a more balanced class ratio (1.7 and 1.2) than the pool data (2.6) it was trained on. It might be worthwhile to explore whether the generation of synthetic training instances can help to mitigate the problem to select instances from both classes in an highly imbalanced data pool.

8 Conclusion

This work is the first to show that Membership Query Synthesis in an NLP setting is feasible. Our approach uses a Variational Autoencoder as a representation learner and generates informative active learning queries from latent space. The classification performance for the generated instances is competitive with pool-based active learning strategies and outperforms other AL strategies with regard to annotation cost (time) and computational complexity.

The main advantage of Membership Query Synthesis for active learning is that it allows us to target specific points along the separating hyperplane and thus to provide the classifier with information on specific areas of uncertainty in the data space. While pool-based active learning has the same objective, Membership Query Synthesis gives us a more precise tool to explore the data space and to generate exactly those instances that we need, making MQS a promising approach for future work in active learning.

Acknowledgments

Part of this research has been conducted within the Leibniz Science Campus "Empirical Linguistics and Computational Modeling", funded by the Leibniz Association under grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art (MWK) of the state of Baden-Württemberg.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *The 57th Conference of the Association for Computational Linguistics*, ACL 2019, pages 6168–6173.

Dana Angluin. 1988. Queries and concept learning. *Machine Learning*, 2(4):319–342.

Pranjal Awasthi and Varun Kanade. 2012. Learning using local membership queries under smooth distributions. *CoRR*, abs/1211.0996.

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364.

Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. 2019. Racial bias in hate speech and abusive language detection datasets. In *The Third Workshop on Abusive Language Online*, pages 25–35.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Tomoki Hayashi, Shinji Watanabe, Yu Zhang, Tomoki Toda, Takaaki Hori, Ramón Fernández Astudillo, and Kazuya Takeda. 2018. Back-translation-style data augmentation for end-to-end ASR. In *2018 IEEE Spoken Language Technology Workshop*, SLT 2018, pages 426–433.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. *CoRR*, abs/1703.00955.

Miriam W. Huijser and Jan C. van Gemert. 2017. Active decision boundary annotation with deep generative models. *CoRR*, abs/1703.06971.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. 2014. Semi-supervised learning with deep generative models. *CoRR*, abs/1406.5298.

Diederik P. Kingma and Max Welling. 2013. Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *CoRR*, abs/1506.06726.

Kevin Lang and Eric Baum. 1992. Query learning can work poorly when a human oracle is used. *IEEE Intl. JointConference on Neural Networks*.

Charles X. Ling and Jun Du. 2008. Active learning with direct query construction. In *The 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 480–487.

Arash Mehrjou, Mehran Khodabandeh, and Greg Mori. 2018. Distributionaware active learning. *arXiv preprint*, abs/1805.08916.

Dai Quoc Nguyen, Dat Quoc Nguyen, Thanh Vu, and Son Bao Pham. 2014. Sentiment classification on polarity reviews: An empirical study using rating-based features. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 128–135.

Augustus Odena. 2016. Semi-supervised learning with generative adversarial networks. *CoRR*, abs/1606.01583.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *CoRR*, abs/1608.05859.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *The 54th Annual Meeting of the Association for Computational Linguistics*, ACL 2016.

Burr Settles. 2009. Active learning literature survey. Technical report, Computer Sciences Technical Report 1648, University of Wisconsin-Madison.

Dinghan Shen, Asli Celikyilmaz, Yizhe Zhang, Liqun Chen, Xin Wang, and Lawrence Carin. 2019. Hierarchically-structured variational autoencoders for long text generation.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Sandeep Subramanian, Adam Trischler, Yoshua Bengio, and Christopher J. Pal. 2018. Learning general purpose distributed sentence representations via large scale multi-task learning. *CoRR*, abs/1804.00079.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Liantao Wang, Xuelei Hu, bo Yuan, and Jianfeng Lu. 2015. Active learning via query synthesis and nearest neighbour search. *Neurocomputing*, 147:426434.

Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. 2019. Detection of abusive language: the problem of biased datasets. In *The 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT 2019, pages 602–608.

Genta Indra Winata, Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2019. Code-switched language models using neural based synthetic data from parallel sentences. In *The SIGNLL Conference on Computational Natural Language Learning*, CoNLL 2019.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational autoencoder for semi-supervised text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Jia-Jie Zhu and José Bento. 2017. Generative adversarial active learning. *CoRR*, abs/1702.07956.

A General-Purpose Algorithm for Constrained Sequential Inference

Daniel Deutsch,[*†] **Shyam Upadhyay,**[*‡◇] **and Dan Roth**[†]
[†]Department of Computer and Information Science, University of Pennsylvania
[‡]Google, New York
{ddeutsch,danroth}@seas.upenn.edu
shyamupa@google.com

Abstract

Inference in structured prediction involves finding the best output structure for an input, subject to certain constraints. Many current approaches use sequential inference, which constructs the output in a left-to-right manner. However, there is no general framework to specify constraints in these approaches. We present a principled approach for incorporating constraints into sequential inference algorithms. Our approach expresses constraints using an *automaton*, which is traversed in lockstep during inference, guiding the search to valid outputs. We show that automata can express commonly used constraints and are easily incorporated into sequential inference. When it is more natural to represent constraints as a set of automata, our algorithm uses an *active set method* for demonstrably fast and efficient inference. We experimentally show the benefits of our algorithm on constituency parsing and semantic role labeling. For parsing, unlike unconstrained approaches, our algorithm always generates valid output, incurring only a small drop in performance. For semantic role labeling, imposing constraints using our algorithm corrects common errors, improving F_1 by 1.5 points. These benefits increase in low-resource settings. Our active set method achieves a 5.2x relative speed-up over a naive approach.[1]

1 Introduction

The key challenge in structured prediction problems (like sequence tagging and parsing) is *inference* (also known as *decoding*), which involves identifying the best output structure $\mathbf{y}$ for an input instance $\mathbf{x}$ from an exponentially large search space $\mathcal{Y}$ (Taskar, 2004; Tsochantaridis et al., 2005). The search is restricted to set of valid $\mathbf{y} \in \mathcal{Y}_\mathbf{x} \subseteq \mathcal{Y}$ for $\mathbf{x}$ by imposing *constraints* during inference. Figure 1 shows examples of constraints used in Semantic Role Labeling (SRL) (Gildea and Jurafsky, 2002).

input:	Alice Smith gave a flower to Bob
gold tags:	B-A0 I-A0 O O B-A1 O B-A2
predicate:	gave
legal args:	A0, A1, A2 (from PropBank)
spans:	[Alice Smith] gave a [flower] to [Bob]
invalid tags:	B-A0 I-A0 O O **B-A0** O B-A2
reason:	duplicate A0
invalid tags:	**B-A5** **I-A5** O O B-A1 O B-A2
reason:	illegal arg A5 for predicate "gave"
invalid tags:	B-A0 **O** O O B-A1 O B-A2
reason:	span [Alice Smith] should have single label

Figure 1: An example SRL instance (input, predicate, gold tags) with different invalid tag sequences and the constraints they violate (details in §4.2).

Currently, inference for most structured prediction problems is solved *sequentially*, predicting the output in a left-to-right manner (Sutskever et al., 2014; Luong et al., 2016). Such *sequential inference* approaches enforce constraints in different ways. For example, a shift-reduce parser consults a stack to determine which action sequences produce valid trees (Nivre et al., 2014), while an SRL model penalizes tag sequences which violate constraints during inference (Punyakanok et al., 2008; Täckström et al., 2015).

At present, inference algorithms are designed to handle task-specific constraints, and there is no general formulation for constrained sequential inference. This contrasts with the state of affairs in NLP before deep learning, when constrained inference approaches used general formulations like Integer Linear Programming (ILP) (Roth and Yih, 2004; Clarke and Lapata, 2008, inter alia).

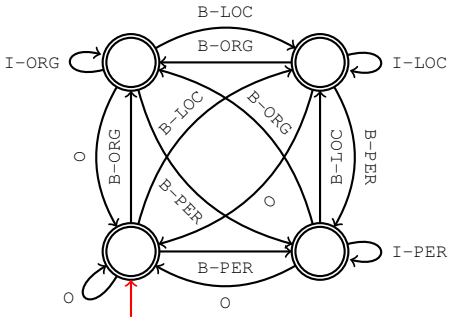
We present a simple, general-purpose sequential inference algorithm that takes a model and

[1]All code available at https://cogcomp.seas.upenn.edu/page/publication_view/884

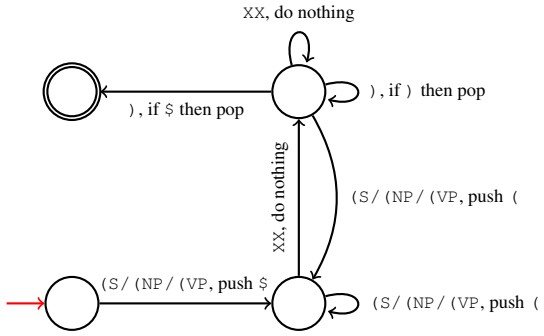
[*]Equal contribution, authors listed alphabetically

[◇]Work done while at University of Pennsylvania.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 482–492
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics



(a) A FSA for BIO tag sequences for NER.



(b) A PDA that accepts valid parse trees like `(S (NP XX) (VP XX XX))` and `(S (VP XX))`. The edge label "P/Q/R, A" denotes consuming either P, Q or R as input and changing the stack per A. "If T" indicates that edge is only valid if T is on top of the stack.

Figure 2: An example FSA and PDA. Red arrows mark start states, and double circles mark accepting states.

an *automaton* expressing the constraints as input, and outputs a structure that satisfies the constraints (§2.2). The automaton guides the inference to always produce a valid output by reshaping the model's probability distribution such that actions deemed invalid by the automaton are not taken.

In some situations, it is more natural to express the constraints as a *set* of automata. However, naively enforcing multiple automata by fully intersecting them is potentially expensive. Instead, our algorithm lazily intersects the automata using an efficient active set method, reminiscent of the cutting-plane algorithm (Tsochantaridis et al., 2005) (§2.4).

The choice of using automata to express constraints has several benefits. First, automata are capable of expressing constraints used in a wide variety of NLP tasks. Indeed, in §3, we show that task-specific constrained inference approaches implicitly use an automaton. Second, automata can be naturally incorporated into any sequential inference algorithm such as beam search. Finally, automata make enforcing multiple constraints straightforward — only the automata for individual constraints need to be specified, which are then intersected at inference time.

Our algorithm is a principled approach for enforcing constraints and has many desirable properties. It decouples the constraints from the inference algorithm, making it generally applicable to many problems. Further, it guarantees valid output and allows for the seamless addition of constraints at inference time without modifying the inference code. We experimentally demonstrate the benefits of our algorithm on two structured prediction tasks, constituency parsing (§5.1) and semantic role labeling (§5.2). Our results in constituency parsing show that our algorithm always outputs valid parses, incurring only a small drop in F_1. In SRL, constrained inference using our algorithm corrects common errors produced by unconstrained inference, resulting in a 1.5 F_1 improvement. This increase in performance is more prominent in low-resource settings. Finally, the active set method for enforcing multiple constraints achieves a 5.2x speed-up over the naive approach of fully intersecting the relevant automata.

2 Constrained Inference with Automata

We briefly review automata that we use for representing constraints in our algorithm.

2.1 Brief Review of Automata Theory

For the purposes of this work, an automaton is a (possibly weighted) directed graph that compactly encodes a set of strings, known as its *language*. The two types of automata used in this work are finite-state automata (FSA) and push-down automata (PDA).[2]

In an FSA, each edge is labeled with a symbol y from an alphabet Σ, and any traversal from the starting state to the final state(s) represents a unique string $\mathbf{y}$ in the language. A PDA is an extension of an FSA in which the traversal can maintain a stack that is used for computation, and the edges may manipulate or examine the state of the stack. Any language that can be expressed using regular expressions or context-free grammars

[2]We only consider deterministic automata.

has an equivalent accepting FSA or PDA, respectively (Sipser, 1997). An example of each type of automata is depicted in Figure 2.

Our inference algorithm views an automaton as an abstract stateful function, denoted as $\mathcal{A}$, which accepts strings from its language $L(\mathcal{A})$. After consuming the prefix $\mathbf{y}_{1:i}$ of a string $\mathbf{y}$, $\mathcal{A}$ provides a score $\mathcal{A}(y_{i+1} \mid \mathbf{y}_{1:i})$ for every symbol $y \in \Sigma$. Invoking $\mathcal{A}$.accepts(s) tests if a string s is in $L(\mathcal{A})$.

2.2 Sequential Inference

The traditional inference problem for structured prediction can be formalized as solving

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}_\mathbf{x}}{\operatorname{argmax}} \log p_\theta(\mathbf{y} \mid \mathbf{x}) \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output structures, $\mathcal{Y}_\mathbf{x}$ is the set of valid output structures for $\mathbf{x}$, and θ are the parameters of the model $p_\theta(\mathbf{y} \mid \mathbf{x})$. A common way to solve this problem is to decompose the objective as follows:

$$\hat{\mathbf{y}} = \underset{\mathbf{y} \in \mathcal{Y}_\mathbf{x}}{\operatorname{argmax}} \sum_i \log p_\theta(y_i \mid \mathbf{x}, \mathbf{y}_{1:i-1}) \tag{2}$$

This decomposition is adopted by popular approaches, such as seq2seq models, SEARN (Daumé et al., 2009), etc. Under this decomposition, the inference problem is often solved by an inexact sequential inference algorithm, such as beam search.

2.3 Imposing Constraints with Automata

We are interested in versions of Equation 2 in which the output space is described by the language of an automaton. Formally,

$$\begin{aligned} \hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} & \sum_i \log p_\theta(y_i \mid \mathbf{x}, \mathbf{y}_{1:i-1}) \\ \text{such that} \quad & \mathbf{y} \in L(\mathcal{A}_\mathbf{x}) \end{aligned} \tag{3}$$

where $L(\mathcal{A}_\mathbf{x})$ is the language of an automaton $\mathcal{A}_\mathbf{x}$ describing the valid output space for instance $\mathbf{x}$. This framework is capable of expressing many constraints which are common in NLP applications, described in detail in §3.

Equation 3 can be rewritten as:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \sum_i \log \tilde{p}_\theta(y_i \mid \mathbf{x}, \mathbf{y}_{1:i-1}) \tag{4}$$

$$\begin{aligned} \log \tilde{p}_\theta(y_i \mid \mathbf{x}, \mathbf{y}_{1:i-1}) \triangleq & \log p_\theta(y_i \mid \mathbf{x}, \mathbf{y}_{1:i-1}) \\ & + \mathcal{A}_\mathbf{x}(y_i \mid \mathbf{y}_{1:i-1}) \end{aligned}$$

Algorithm 1 Constrained Sequential Greedy Inference

Input: Model p_θ, automaton $\mathcal{A}_\mathbf{x}$, sequence $\mathbf{x}$
Output: Prediction $\hat{\mathbf{y}} \in L(\mathcal{A}_\mathbf{x})$

1: **procedure** CONSTRAINED-SEARCH(p_θ, $\mathcal{A}_\mathbf{x}$, $\mathbf{x}$)
2: $\quad\hat{\mathbf{y}} \leftarrow$ [BOS] ▷ *Beginning of sequence token*
3: $\quad$**while** $\hat{\mathbf{y}}$ is not finished **do**
4: $\quad\quad\log \tilde{p}_\theta(y' \mid \mathbf{x}, \hat{\mathbf{y}}) = \log p_\theta(y' \mid \mathbf{x}, \hat{\mathbf{y}}) + \mathcal{A}_\mathbf{x}(y' \mid \hat{\mathbf{y}})$
5: $\quad\quad y \leftarrow \arg\max_{y'} \log \tilde{p}_\theta(y' \mid \mathbf{x}, \hat{\mathbf{y}})$
6: $\quad\quad\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} + y$ ▷ *Extend current sequence*
7: $\quad$**return** $\hat{\mathbf{y}}$

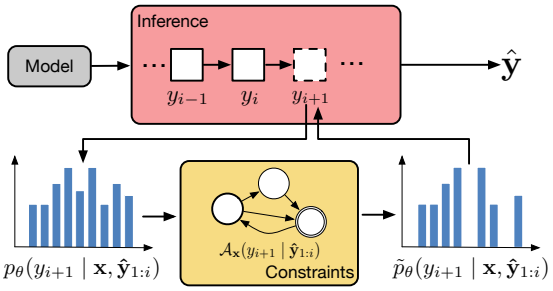


Figure 3: At each time step, the automata $\mathcal{A}_\mathbf{x}$ reshapes the model's probability distribution p_θ to $\tilde{p}_\theta$, from which the next output label y_{i+1} is determined.

where $\mathcal{A}_\mathbf{x}$ reshapes p_θ to $\tilde{p}_\theta$ at each time step. To impose hard constraints, we set this score to $-\infty$ for invalid y_i and constant for all valid y_i.[3] Equation 4 allows natural extensions of sequential inference algorithms, where an automaton is traversed in lock-step to guide the search to always output a valid structure. The necessary extension of greedy search is presented in Algorithm 1. Concretely, when predicting y_i, the inference selects the most probable action under the reshaped probability distribution $\tilde{p}_\theta$ (line 5). It is straightforward to similarly extend other sequential inference algorithms.

2.4 Imposing Multiple Constraints

The formulation above assumes that the constraints are described using a single automaton. However, in some scenarios, it is more natural to impose multiple constraints by representing them as a set of automata.

A Motivating Example. Consider Figure 1 which illustrates the SRL problem. The following constraints should be obeyed by the tag sequence: the predicate cannot have multiple arguments of the same type ($\texttt{A}_0$–$\texttt{A}_5$), each predicate may only accept a certain set of argument types, and certain spans derived from a constituency parse should re-

[3] Soft constraints can also be imposed, but are not explored in this work.

ceive the same label.[4] Each of these constraints can be easily expressed using a separate automaton. In contrast, directly writing a single automaton to enforce these constraints simultaneously could be impractical and difficult.

Issues with Naive Approaches. Naively extending sequential inference algorithms to impose a set of constraints by traversing multiple automata in parallel fails. There is no guarantee that a valid structure will be found, even if the intersection of the automata's languages is non-empty (a proof by counter-example is provided in Appendix A). The alternative solution of intersecting the automata into a single automaton may be intractable, as the size of the intersected automaton grows exponentially in the number of constraints (Hopcroft and Ullman, 1979).

An Active Set Method. Intuitively, intersecting all of the automata may not be necessary because it is possible for a constraint to be satisfied without it being enforced. This is the basis for *active set* methods (such as the cutting-plane algorithm (Kelley, 1960; Tsochantaridis et al., 2005)), which maintain a set during inference that contains currently active (i.e., enforced) constraints. When a constraint is violated by the current output, it enters (i.e., is added to) the active set.

We present an active set method for imposing multiple constraints represented by a set of automata $\mathcal{S}_{\mathbf{x}}$ in Algorithm 2. Our algorithm is inspired by the active set algorithm of Tromble and Eisner (2006) for finite-state transducers.

For an instance $\mathbf{x}$, Algorithm 2 maintains a active set[5] $\mathcal{W}$ corresponding to all violated constraints so far. $\mathcal{W}$ is represented by the intersection $\mathcal{A}_{\mathcal{W}}$ of the relevant automata, which is initialized with an automata Σ^* that accepts any sequence (line 1). On each iteration, the algorithm runs a constrained inference algorithm (such as Algorithm 1) that uses $\mathcal{A}_{\mathcal{W}}$ (line 3) to find an output $\hat{\mathbf{y}}$. Then, FIND-VIOLATION checks if $\hat{\mathbf{y}}$ violates any of the constraints that are not currently in the active set, $\mathcal{S}_{\mathbf{x}} \setminus \mathcal{W}$ (line 4). If $\hat{\mathbf{y}}$ is accepted by all of the automata (line 5), it is valid and subsequently returned (line 6). Otherwise, the first violated constraint is added to $\mathcal{W}$ (line 8), its automaton $\mathcal{A}'$ intersected with $\mathcal{A}_{\mathcal{W}}$ (line 9), and constrained inference is re-run (line 3).

[4]These constraints are described in detail in §4.

[5]Also known as a *working set*.

Algorithm 2 An Active Set Method for Multiple Constraints

Input: Model p_θ, set of automata $\mathcal{S}_{\mathbf{x}}$, sequence $\mathbf{x}$
Output: Prediction $\hat{\mathbf{y}} \in \bigcap_{\mathcal{A} \in \mathcal{S}_{\mathbf{x}}} L(\mathcal{A})$

1: $\mathcal{W} \leftarrow \emptyset$, $\mathcal{A}_{\mathcal{W}} \leftarrow \Sigma^*$
2: **while** True **do**
3: $\quad \hat{\mathbf{y}} \leftarrow$ CONSTRAINED-SEARCH(p_θ, $\mathcal{A}_{\mathcal{W}}$, $\mathbf{x}$)
4: $\quad \mathcal{A}' \leftarrow$ FIND-VIOLATION($\mathcal{S}_{\mathbf{x}} \setminus \mathcal{W}$, $\hat{\mathbf{y}}$)
5: $\quad$ **if** $\mathcal{A}'$ is null **then** ▷ *No constraint is violated*
6: $\quad\quad$ **return** $\hat{\mathbf{y}}$ ▷ *return current output*
7: $\quad$ **else**
8: $\quad\quad \mathcal{W} \leftarrow \mathcal{W} \cup \{\mathcal{A}'\}$ ▷ *update working set*
9: $\quad\quad \mathcal{A}_{\mathcal{W}} \leftarrow \mathcal{A}_{\mathcal{W}} \cap \mathcal{A}'$ ▷ *automata intersection*
10: **procedure** FIND-VIOLATION(K, $\mathbf{y}$)
11: $\quad$ **for** each $\mathcal{A}$ in K **do**
12: $\quad\quad$ **if** not $\mathcal{A}$.accepts($\mathbf{y}$) **then**
13: $\quad\quad\quad$ **return** $\mathcal{A}$ ▷ *the first violated constraint*
14: $\quad$ **return** null

Algorithm 2 is guaranteed to terminate with a valid output. In the worst case, all of the constraints will be eventually enter the active set and inference will run with a fully intersected automata. Although the cost of this worst case is exponential in the number of constraints, this occurs infrequently in practice. Moreover, we found that Algorithm 2 is faster than naively computing the full intersection despite running inference multiple times.

3 Representing Constraints as Automata

We now illustrate the expressibility of automata by showing how they can represent commonly used constraints in various NLP applications.

Text Generation. Text generation tasks like image captioning, machine translation, sentence simplification, etc., often require that the output must contain specific words or phrases (Anderson et al., 2017; Hokamp and Liu, 2017; Post and Vilar, 2018; Zhang et al., 2017) in order to incorporate prior knowledge (e.g., the caption must have the word "chair" as the object was detected in the image). Similarly, constraints can disallow invalid sequences, such as words which do not rhyme (Ghazvininejad et al., 2016, 2018) or do not appear in a dictionary (Deutsch et al., 2018). These constraints can be represented as FSAs where all paths to an accepting state contain the required sequences and do not contain any disallowed sequences. Note that the size of the automata is not a function of the output vocabulary but the vocabulary that participates in the constraints.

Sequence Tagging. Many sequence tagging problems (such as NER, shallow parsing, etc.) require that the output tags are valid under the specific tagging scheme, such as BIO, which marks each token as beginning, inside, or outside of a phrase. One can easily write an FSA that recognizes the language of valid tag sequences for these schemes, such as the automaton in Figure 2a.

Other constraints commonly applied to sequence tagging are specific to the particular task. In SRL, each argument type can appear exactly one time. For instance, for the label $\texttt{A}_0$, an FSA with 3 states (before-$\texttt{A}_0$, emitting-$\texttt{A}_0$ and after-$\texttt{A}_0$) can be written to enforce this constraint. See Tromble and Eisner (2006) for examples.

Syntactic Parsing. Syntactic parsing (dependency or constituency) tasks require that the output forms a valid tree, a constraint commonly enforced using the shift-reduce algorithm (Zhu et al., 2013; Nivre et al., 2014; Dyer et al., 2016). Shift-reduce inference inspects the state of a stack to decide which next actions are valid (e.g., if the stack is empty, reduce is invalid). Shift-reduce inference is implicitly using an automaton which is the intersection of a PDA (that counts how many shift and reduce actions have occurred) and an FSA (that restricts the maximum number of actions based on the input sentence length).

Semantic Parsing and Code Generation. In semantic parsing and code generation, constraints ensure both the syntactic validity and the executability of the output. For instance, for the predicate ISADJACENT (which compares two countries for adjacency) the syntactic constraint ensures the predicate receives two arguments, whereas the executability ensures they are properly typed (e.g., that they are countries). Krishnamurthy et al. (2017) use a type-constrained grammar to ensure the output logical form is well-typed. Similarly, Ling et al. (2016) use a stack while decoding to ensure the validity of the generated code. Both these constraints can be encoded in a PDA derived from the grammar of the language (Sipser, 1997).

4 Experimental Setup

We elaborate on two constrained inference tasks from the previous section, constituency parsing and semantic role labeling, which serve as case studies for showing the applicability and versatility of our approach. Our goal is not to beat the state-of-the-art, but to illustrate the practicality and benefits of our approach.

4.1 Models, Datasets, and Evaluation

Constituency Parsing. We follow the experimental setup of Vinyals et al. (2015) and train a seq2seq model with attention using standard splits on the Penn Treebank. This setup was chosen to ensure that no constraints are built into the model for a truly unconstrained baseline. The input is a sequence of tokens, and the model outputs a linearized parse tree ("gold parse" in Figure 4).

We compare our approach (CONSTRAINED) to unconstrained inference (UNCONSTRAINED), which runs beam search and selects the highest-scoring output. We also compare to Vinyals et al. (2015)'s inference approach (henceforth POSTHOC). Vinyals et al. (2015) removes parses without the correct number of preterminals at the end of beam search and adds parentheses to almost-valid parses.[6] The algorithms are evaluated using F_1 as reported by EVALB, a standard evaluation toolkit.[7] However, because EVALB ignores invalid trees during evaluation (which can artificially improve the performance of models which violate constraints), we also report coverage, the percentage of valid outputs. We use a beam size of 10.

Semantic Role Labeling. For SRL, the unconstrained model is an off-the-shelf implementation of He et al. (2017). The input to the model is the sentence and the predicate for which the arguments need to be identified, and the model outputs a tag sequence (such as Figure 1). To isolate the effect of constraints, we assume the gold predicates are available, unlike He et al. (2017). We use the standard train and development splits from the CoNLL 2005 shared task and the same model parameters as He et al. (2017). We report the CoNLL F_1 score[8] computed for the core arguments (namely $\texttt{A}_0$–$\texttt{A}_5$) and use greedy inference.

4.2 Constraints

Constituency Parsing. The constraints used in parsing disallow invalid parses, such as the examples in Figure 4. The constraints ensure that **(i)** the parentheses are balanced (BAL), **(ii)** there is exactly one preterminal per input token

[6] personal communication with Vinyals et al. (2015)
[7] nlp.cs.nyu.edu/evalb/
[8] www.lsi.upc.es/~srlconll/srl-eval.pl

input:	John kissed Mary
gold parse:	`(S (NP XX) (VP XX (NP XX)))`
invalid parse: **reason**:	`(S (NP ) (VP XX XX (NP XX)))` empty phrase
invalid parse: **reason**:	`(S (VP XX (NP XX)))` incorrect number of preterminals
invalid parse: **reason**:	`(S (NP XX) (VP XX (NP XX)))` unbalanced parentheses

Figure 4: An example parsing instance with different invalid parses and the constraints they violate (details in §4.2). XX denotes POS tags.

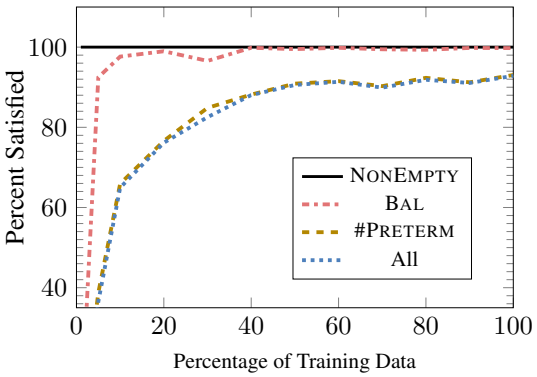


Figure 5: Percentage of output test parses that satisfy the constituency parsing constraints in §4.2 when using unconstrained inference as the amount of training data is increased. "All" measures the percent of parses which satisfy all of the constraints. Even with 100% of the training data, the model fails to always output the correct number of preterminals.

(#Preterm), and **(iii)** every phrase contains at least one preterminal (NonEmpty).

Semantic Role Labeling. The constraints used in SRL disallow invalid sequences, such as the ones in Figure 1. The constraints ensure that **(i)** a predicate cannot have multiple arguments of the same type (NoDup). This constraint is expressed using one FSA per argument label for a total of 5 automata; **(ii)** a predicate can only take a certain set of argument types (LegalArgs), according to PropBank v3.1 (Palmer et al., 2005). This constraint is expressed using one FSA; **(iii)** certain spans in the sentence should be assigned the same argument label type (SpanLabel). These spans are identified from the predicate and the constituency parse of the sentence using the heuristic of Xue and Palmer (2004).[9] This constraint is expressed using one FSA.

Ease of Implementation. All of the constraints were expressed using an FSA, with the exception of Bal which requires a PDA. The automata were implemented using Pynini (Gorman, 2016). We use the same inference code for both tasks, except for the automata generation.

5 Experimental Results

We show the practicality of our approach and the benefits over unconstrained inference in the experiments below. In order to illustrate the benefits of constrained inference in low-resource settings, we simulate different levels of supervision.

5.1 Constituency Parsing

We experimentally show the need for constraints and compare inference strategies for parsing.

Necessity of Constraints. In order to demonstrate that constraints are necessary to guarantee structurally valid output, we vary the amount of training data for a seq2seq model and measure how frequently the output is an invalid parse tree.

The results in Figure 5 show that while the model learns some constraints with little data, others are frequently violated even with full supervision. For instance, the model outputs parse trees with balanced parentheses for over 94% of instances, after training on as little as 5% of the training data. However, the model frequently violates the #Preterm constraint, outputting an incorrect number of preterminals on about 7% of parses even after training on 100% of the data. Therefore, we cannot expect seq2seq models to learn to produce valid outputs, even when it has access to 40k labeled parse trees, let alone in low-resource settings.

Comparing Inference Approaches. Figure 6 shows the performance of all the three inference techniques discussed in §4.1, namely, Unconstrained, PostHoc, and Constrained.

At first glance, it appears that Unconstrained is slightly better than both PostHoc and Constrained at 100% supervision. However, EVALB ignores any invalid parse trees when computing F_1, and therefore it is necessary to take coverage (the percent of valid output parses) into account when comparing inference approaches. It is evident from Figure 6 (left) that Constrained always ensures 100% coverage, and PostHoc only reaches near-100% coverage with full supervision. In contrast, Unconstrained

[9] We use Kitaev and Klein (2018)'s parser.

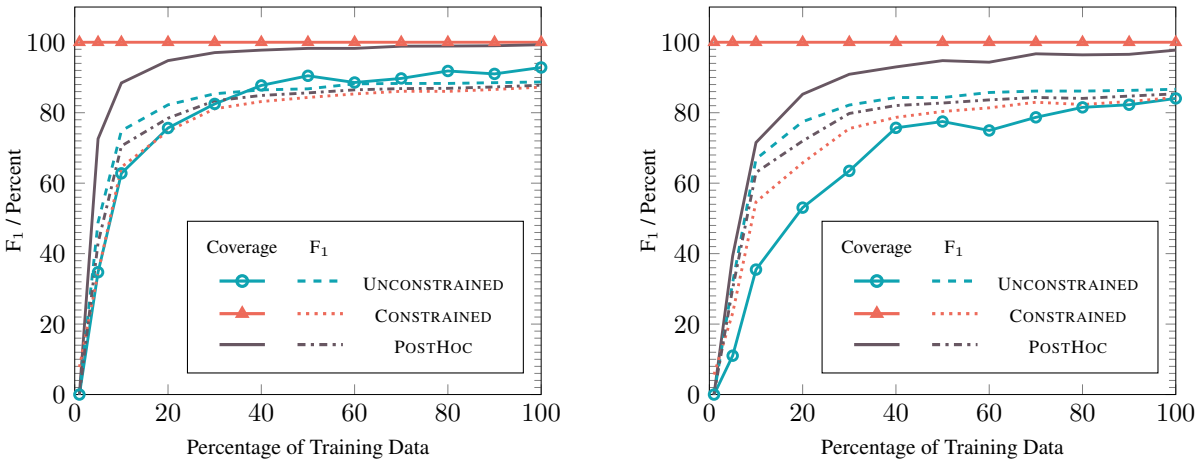


Figure 6: F_1 and coverage (# valid output parses / # test instances) for UNCONSTRAINED, POSTHOC, and CONSTRAINED on the entire test data (**left**) and test sentences of length ≥30 (**right**). The UNCONSTRAINED and POSTHOC inference algorithms have a harder time producing valid parse trees for the longer input sentences.

achieves consistently low coverage, with the best model reaching 7% lower coverage than the CONSTRAINED.

The increased coverage explains the apparent drop in F_1 for constrained approaches. Intuitively, longer sentences are inherently harder to parse. CONSTRAINED and POSTHOC produce a valid, but potentially incorrect parse for these sentences and get penalized. In contrast, UNCONSTRAINED is more likely to produce invalid parses for these sentences, and is thus effectively evaluated on a test set containing short sentences. To verify this, we re-evaluated the inference approaches on test sentences which are ≥30 tokens (Figure 6, right). Under this setting, every approach has worse F_1 and coverage at all levels of supervision (Figure 6 left vs right).

5.2 Semantic Role Labeling

We show that we can improve the performance of a trained model by incorporating constraints at inference time which address common errors (Daza and Frank, 2018) made by SRL models. This experiment also shows the efficiency of the active set method in enforcing multiple constraints.

Correcting Common Errors. We now show how some common SRL errors like the ones discussed in Figure 1, can be corrected using constraints added at inference time. Starting with an unconstrained baseline model (UNCONSTRAINED), we successively add the NODUP, LEGALARGS and SPANLABEL constraints, in that order.

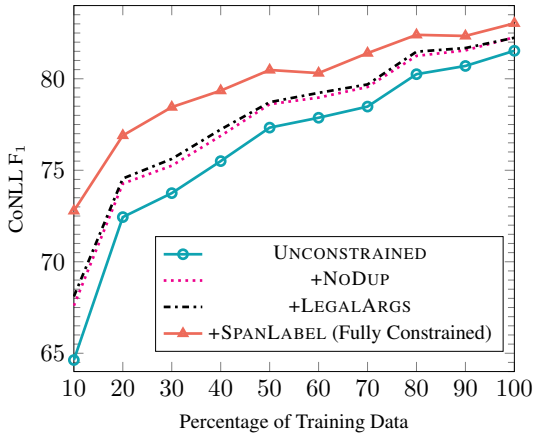


Figure 7: Learning curve for SRL. Starting with unconstrained inference, we incrementally add NODUP, LEGALARGS and SPANLABEL constraints. The constraints improve performance at all levels of supervision, with the largest improvements in low supervision settings.

Figure 7 shows that using the constraints improves performance over UNCONSTRAINED in all settings. Constraints like NODUP and SPANLABEL give significant gains, demonstrating their value. At 100% supervision, fully constrained inference improves by 1.5 F_1 points over unconstrained inference (83.03 vs 81.53). In fact, constrained inference at only 50% data achieves similar F_1 score as UNCONSTRAINED at 100%, with larger gains in low supervision regimes (≤40%).

Active Set Size and Efficiency. For SRL, the maximum possible size of the active set $\mathcal{W}$ is 7 for any test instance. Intersecting all 7 automata would lead to an automaton with 1043 states and

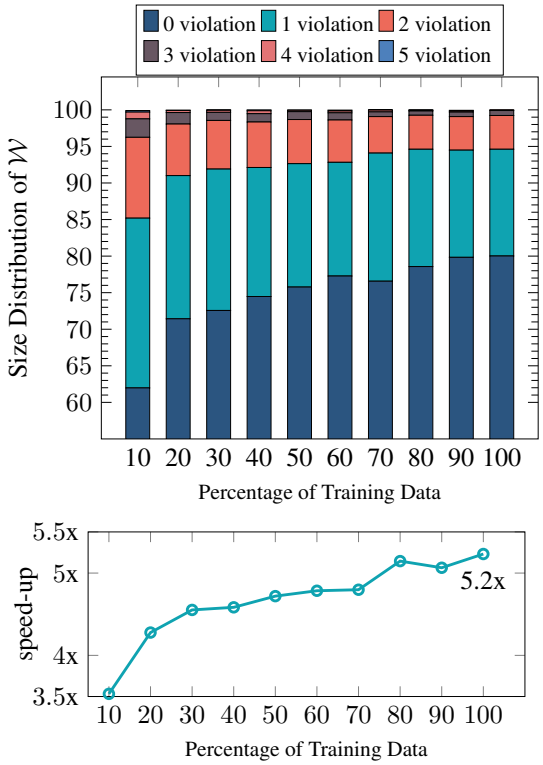


Figure 8: **Top**: Size distribution of the final working set for the active set method. With a model trained on 30% of the data, 72% of the test instances have an empty final working set (i.e., with 0 violations). **Bottom**: Relative speed-up in decoding time using the active set method over naively computing the full intersection.

2022 arcs. We now measure the size of the active set observed in practice.

Figure 8 shows the size distribution of the final $\mathcal{W}$ under different amounts of supervision. With a fully supervised model, 80% of the test instances had an empty $\mathcal{W}$ when inference terminated (i.e., no constraints entered $\mathcal{W}$). Under the same setting, $\mathcal{W}$ contains 1 and 2 constraints for 10.6% and 8.6% of the instances, respectively. Under any setting, $\mathcal{W}$ was empty for >60% of the instances. On average, the active set automaton had 34 states and 66 arcs, a 30x reduction over the full intersection.

To evaluate the efficiency of Algorithm 2, Figure 8 (bottom) plots the relative decoding times of the active set method over naively computing the full intersection. The active set method is consistently faster, with the largest speed-up (5.2x) at the highest level of supervision as the average active set size is the smallest at this setting.

Factors Affecting Speed-up. In general, the amount of speed-up provided by the active set method depends on several factors, including the number of constraints, the size of the constraint automata, and the cost of computing the softmax during inference. The largest gains will come when the former two factors are most expensive, as the active set will only incur the intersection cost as needed. If the output vocabulary is large, softmax computation may outweigh the cost of fully intersecting the constraint automata.

6 Related Work

Traditional Constrained Inference. Traditional constrained inference approaches enforced constraints using general combinatorial optimization frameworks, for instance linear (Taskar et al., 2004) and integer linear programs (Roth and Yih, 2005, 2007; Clarke and Lapata, 2008; Martins et al., 2009), SAT solvers (Richardson and Domingos, 2006; Poon and Domingos, 2008), etc. Unlike these approaches which find the best output that satisfies a set of constraints, our work attempts to provide a similar general framework for sequential inference algorithms that will find the approximate-best output.

Unconstrained Data-driven Approaches. Many sequential inference approaches do not enforce constraints at all, in the hope that they will be learned from data (Lample et al., 2016; Choe and Charniak, 2016; Suhr et al., 2018). While the model can potentially "impose" some constraints which are well-represented in the data, there is no guarantee that the output structure will be valid. In contrast, our work guarantees valid output.

Post-Hoc Constraint Satisfaction. Some approaches first run unconstrained inference to find the top-k structures and then identify valid structures in a post-hoc manner (Andreas et al., 2013; Vinyals et al., 2015; Kiddon et al., 2016; Upadhyay et al., 2018). Such techniques also cannot guarantee validity of the output structure when all top-k structures are invalid, whereas our model ensures all top-k are valid.

Our Work. We draw from work in NLP that uses automata (Mohri, 1997; Karttunen, 2000, inter alia) and recent work in constrained text generation (discussed in §3). Our work is also related to Anderson et al. (2017) who impose lexical constraints for image captioning by maintaining a beam for *each* state in an FSA, an impractical strategy for automata with thousands of states (like those in §5.2). Tromble and Eisner (2006)

was the inspiration for the active set method. A very recent work, Lee et al. (2019), also impose constraints in sequential inference by including them in the objective function and modifying the model's weights until the constraints are satisfied.

7 Conclusion and Future Work

We presented a principled, general-purpose constrained sequential inference algorithm. Key to our algorithm is using automata to represent constraints, which we showed are capable of expressing popularly used constraints in NLP. Our approach is an attractive alternative to task-specific constrained inference approaches currently in use. Using a fast active set method, we can seamlessly incorporate multiple constraints at inference time without modifying the inference code. The experimental results showed the value of our approach over unconstrained inference, with the gains becoming more prominent in low-resource settings.

Acknowledgments

The authors would like to thank Sampath Kannan, Caleb Stanford, Jordan Kodner, and the anonymous reviewers for their useful comments and suggestions.

This work was supported by Contract HR0011-15-C-0113 and Contract HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided Open Vocabulary Image Captioning with Constrained Beam Search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 936–945, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic Parsing as Machine Translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria. Association for Computational Linguistics.

Do Kook Choe and Eugene Charniak. 2016. Parsing as Language Modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based Structured Prediction. *Machine learning*, 75(3):297–325.

Angel Daza and Anette Frank. 2018. A Sequence-to-Sequence Model for Semantic Role Labeling. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 207–216. Association for Computational Linguistics.

Daniel Deutsch, John Hewitt, and Dan Roth. 2018. A Distributional and Orthographic Aggregation Model for English Derivational Morphology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1938–1947. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Marjan Ghazvininejad, Yejin Choi, and Kevin Knight. 2018. Neural Poetry Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 67–71, New Orleans, Louisiana. Association for Computational Linguistics.

Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. 2016. Generating Topical Poetry. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, Austin, Texas. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

Kyle Gorman. 2016. Pynini: A Python Library for Weighted Finite-state Grammar Compilation. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, pages 75–80, Berlin, Germany. Association for Computational Linguistics.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep Semantic Role Labeling: What Works and What's Next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages

473–483. Association for Computational Linguistics.

Chris Hokamp and Qun Liu. 2017. Lexically Constrained Decoding for Sequence Generation Using Grid Beam Search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

John E Hopcroft and Jeffrey D Ullman. 1979. *Introduction to automata theory, languages, and computation*. Addison-Wesley.

Lauri Karttunen. 2000. Applications of finite-state transducers in natural language processing. In *International Conference on Implementation and Application of Automata*, pages 34–46. Springer.

James E Kelley, Jr. 1960. The Cutting-Plane Method for Solving Convex Programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally Coherent Text Generation with Neural Checklist Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686. Association for Computational Linguistics.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Jay Yoon Lee, Sanket Vaibhav Mehta, Michael Wick, Jean-Baptiste Tristan, and Jaime G. Carbonell. 2019. Gradient-based inference for networks with output constraints. In *AAAI 2019*.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609. Association for Computational Linguistics.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task Sequence to Sequence Learning. In *Proc. of ICLR*.

André FT Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise Integer Linear Programming Formulations for Dependency Parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Mehryar Mohri. 1997. Finite-state Transducers in Language and Speech Processing. *Computational linguistics*, 23(2):269–311.

Joakim Nivre, Yoav Goldberg, and Ryan McDonald. 2014. Constrained Arc-Eager Dependency Parsing. *Computational Linguistics*, 40(2).

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Hoifung Poon and Pedro Domingos. 2008. Joint Unsupervised Coreference Resolution with Markov Logic. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 650–659.

Matt Post and David Vilar. 2018. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324. Association for Computational Linguistics.

Vasin Punyakanok, Dan Roth, and Wen tau Yih. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics*, 34(2).

Matthew Richardson and Pedro Domingos. 2006. Markov Logic Networks. *Machine Learning Journal*, 62(1-2):107–136.

Dan Roth and Scott Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics.

Dan Roth and Scott Wen-tau Yih. 2005. Integer Linear Programming Inference for Conditional Random Fields. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 737–744.

Dan Roth and Wen-tau Yih. 2007. Global Inference for Entity and Relation Identification via a Linear Programming Formulation.

Michael Sipser. 1997. *Introduction to the Theory of Computation*, volume 2. PWS Publishing.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to Map Context-Dependent Sentences to Executable Formal Queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2238–2249. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

B. Taskar. 2004. *Learning Structured Prediction Models: A Large Margin Approach*. Ph.D. thesis, Stanford University. Computer Science TR-280.

B. Taskar, C. Guestrin, and D. Koller. 2004. Max-margin markov networks. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*.

Roy Tromble and Jason Eisner. 2006. A fast finite-state relaxation method for enforcing global constraints on sequence decoding. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 423–430, New York City, USA. Association for Computational Linguistics.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.

Shyam Upadhyay, Jordan Kodner, and Dan Roth. 2018. Bootstrapping Transliteration with Constrained Discovery for Low-Resource Languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 501–511. Association for Computational Linguistics.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a Foreign Language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

Nianwen Xue and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*, pages 88–94, Barcelona, Spain.

Yaoyuan Zhang, Zhenxu Ye, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. A Constrained Sequence-to-Sequence Neural Model for Sentence Simplification. *CoRR*, abs/1704.02312.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, Sofia, Bulgaria. Association for Computational Linguistics.

A Richly Annotated Corpus for Different Tasks in Automated Fact-Checking

Andreas Hanselowski[†*], Christian Stab[†*], Claudia Schulz[†*], Zile Li[*], Iryna Gurevych[†*]

[†]Research Training Group AIPHES
https://www.aiphes.tu-darmstadt.de

[*]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
https://www.ukp.tu-darmstadt.de/

[†*] Computer Science Department, Technische Universität Darmstadt

Abstract

Automated fact-checking based on machine learning is a promising approach to identify false information distributed on the web. In order to achieve satisfactory performance, machine learning methods require a large corpus with reliable annotations for the different tasks in the fact-checking process. Having analyzed existing fact-checking corpora, we found that none of them meets these criteria in full. They are either too small in size, do not provide detailed annotations, or are limited to a single domain. Motivated by this gap, we present a new substantially sized mixed-domain corpus with annotations of good quality for the core fact-checking tasks: document retrieval, evidence extraction, stance detection, and claim validation. To aid future corpus construction, we describe our methodology for corpus creation and annotation, and demonstrate that it results in substantial inter-annotator agreement. As baselines for future research, we perform experiments on our corpus with a number of model architectures that reach high performance in similar problem settings. Finally, to support the development of future models, we provide a detailed error analysis for each of the tasks. Our results show that the realistic, multi-domain setting defined by our data poses new challenges for the existing models, providing opportunities for considerable improvement by future systems.

1 Introduction

The ever-increasing role of the Internet as a primary communication channel is arguably the single most important development in the media over the past decades. While it has led to unprecedented growth in information coverage and distribution speed, it comes at a cost. False information can be shared through this channel reaching a much wider audience than traditional means of disinformation (Howell et al., 2013).

While human fact-checking still remains the primary method to counter this issue, the amount and the speed at which new information is spread makes manual validation challenging and costly. This motivates the development of automated fact-checking pipelines (Thorne et al., 2018a; Popat et al., 2017; Hanselowski and Gurevych, 2017) consisting of several consecutive tasks. The following four tasks are commonly included in the pipeline. Given a controversial claim, *document retrieval* is applied to identify documents that contain important information for the validation of the claim. *Evidence extraction* aims at retrieving text snippets or sentences from the identified documents that are related to the claim. This evidence can be further processed via *stance detection* to infer whether it supports or refutes the claim. Finally, *claim validation* assesses the validity of the claim given the evidence.

Automated fact-checking has received significant attention in the NLP community in the past years. Multiple corpora have been created to assist the development of fact-checking models, varying in quality, size, domain, and range of annotated phenomena. Importantly, the successful development of a full-fledged fact-checking system requires that the underlying corpus satisfies certain characteristics. First, training data needs to contain a large number of instances with *high-quality annotations* for the different fact-checking sub-tasks. Second, the training data should not be limited to a particular domain, since potentially wrong information sources can range from official statements to blog and Twitter posts.

We analyzed existing corpora regarding their adherence to the above criteria and identified several drawbacks. The corpora introduced by Vlachos and Riedel (2014); Ferreira and Vlachos (2016); Derczynski et al. (2017) are valuable for the analysis of the fact-checking problem and pro-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 493–503
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

vide annotations for stance detection. However, they contain only several hundreds of validated claims and it is therefore unlikely that deep learning models can generalize to unobserved claims if trained on these datasets.

A corpus with significantly more validated claims was introduced by Popat et al. (2017). Nevertheless, for each claim, the corpus provides 30 documents which are retrieved from the web using the Google search engine instead of a document collection aggregated by fact-checkers. Thus, many of the documents are unrelated to the claim and important information for the validation may be missing.

The FEVER corpus constructed by Thorne et al. (2018a) is the largest corpus available for the development of automated fact-checking systems. It consists of 185,445 validated claims with annotated documents and evidence for each of them. The corpus therefore allows training deep neural networks for automated fact-checking, which reach higher performance than shallow machine learning techniques. However, the corpus is based on synthetic claims derived from Wikipedia sentences rather than *natural* claims that originate from heterogeneous web sources.

In order to address the drawbacks of existing datasets, we introduce a new corpus based on the Snopes[1] fact-checking website. Our corpus consists of 6,422 validated claims with comprehensive annotations based on the data collected by Snopes fact-checkers and our crowd-workers. The corpus covers multiple domains, including discussion blogs, news, and social media, which are often found responsible for the creation and distribution of unreliable information. In addition to validated claims, the corpus comprises over 14k documents annotated with evidence on two granularity levels and with the stance of the evidence with respect to the claims. Our data allows training machine learning models for the four steps of the automated fact-checking process described above: document retrieval, evidence extraction, stance detection, and claim validation.

The contributions of our work are as follows:

1) We provide a substantially sized mixed-domain corpus of natural claims with annotations for different fact-checking tasks. We publish a web crawler that reconstructs our dataset including all annotations[2]. For research purposes, we are allowed to share the original corpus[3].

2) To support the creation of further fact-checking corpora, we present our methodology for data collection and annotation, which allows for the efficient construction of large-scale corpora with a substantial inter-annotator agreement.

3) For evidence extraction, stance detection, and claim validation we evaluate the performance of high-scoring systems from the FEVER shared task (Thorne et al., 2018b)[4] and the Fake News Challenge (Pomerleau and Rao, 2017)[5] as well as the Bidirectional Transformer model BERT (Devlin et al., 2018) on our data. To facilitate the development of future fact-checking systems, we release the code of our experiments[6].

4) Finally, we conduct a detailed error analysis of the systems trained and evaluated on our data, identifying challenging fact-checking instances which need to be addressed in future research.

2 Related work

Below, we give a comprehensive overview of existing fact-checking corpora, summarized in Table 1. We focus on their key parameters: fact-checking sub-task coverage, annotation quality, corpus size, and domain. It must be acknowledged that a fair comparison between the datasets is difficult to accomplish since the length of evidence and documents, as well as the annotation quality, significantly varies between the corpora.

PolitiFact14 Vlachos and Riedel (2014) analyzed the fact-checking problem and constructed a corpus on the basis of the fact-checking blog of Channel 4[7] and the Truth-O-Meter from PolitiFact[8]. The corpus includes additional evidence, which has been used by fact-checkers to validate the

[1] http://www.snopes.com/

[2] https://github.com/UKPLab/conll2019-snopes-crawling

[3] We crawled and provide the data according to the regulations of the German text and data mining policy. That is, the crawled documents/corpus may be shared upon request with other researchers for non-commercial purposes through the research data archive service of the university library. Please request the data at https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/2081

[4] http://fever.ai/task.html/

[5] http://www.fakenewschallenge.org/

[6] https://github.com/UKPLab/conll2019-snopes-experiments

[7] http://blogs.channel4.com/factcheck/

[8] http://www.politifact.com/truth-o-meter/statements/

	claims	docs.	evid.	stance	sources	rater agr.	domain
PolitiFact14	106	no	yes	no	no	no	political statements
Emergent16	300	2,595	no	yes	yes	no	news
PolitiFact17	12,800	no	no	no	no	no	political statements
RumourEval17	297	4,519	no	yes	yes	yes	Twitter
Snopes17	4,956	136,085	no	no	yes	no	Google search results
CLEF-2018	150	no	no	no	no	no	political debates
FEVER18	185,445	14,533	yes	yes	yes	yes	Wikipedia
Our corpus	6,422	14,296	yes	yes	yes	yes	multi domain

Table 1: Overview of corpora for automated fact-checking. docs: documents related to the claims; evid.: evidence in form of sentence or text snippets; stance: stance of the evidence; sources: sources of the evidence; rater agr.: whether or not the inter-annotator agreement is reported; domain: the genre of the corpus

claims, as well as metadata including the speaker ID and the date when the claim was made. This is early work in automated fact-checking and Vlachos and Riedel (2014) mainly focused on the analysis of the task. The corpus therefore only contains 106 claims, which is not enough to train high-performing machine learning systems.

Emergent16 A more comprehensive corpus for automated fact-checking was introduced by Ferreira and Vlachos (2016). The dataset is based on the project Emergent[9] which is a journalist initiative for rumor debunking. It consists of 300 claims that have been validated by journalists. The corpus provides 2,595 news articles that are related to the claims. Each article is summarized into a headline and is annotated with the article's stance regarding the claim. The corpus is well suited for training stance detection systems in the news domain and it was therefore chosen in the Fake News Challenge (Pomerleau and Rao, 2017) for training and evaluation of competing systems. However, the number of claims in the corpus is relatively small, thus it is unlikely that sophisticated claim validation systems can be trained using this corpus.

PolitiFact17 Wang (2017) extracted 12,800 validated claims made by public figures in various contexts from Politifact. For each statement, the corpus provides a verdict and meta information, such as the name and party affiliation of the speaker or subject of the debate. Nevertheless, the corpus does not include evidence and thus the models can only be trained on the basis of the claim, the verdict, and meta information.

RumourEval17 Derczynski et al. (2017) organized the RumourEval shared task, for which they provided a corpus of 297 rumourous threads from Twitter, comprising 4,519 tweets. The shared task was divided into two parts, *stance detection* and *veracity prediction* of the rumors, which is similar to claim validation. The large number of stance-annotated tweets allows for training stance detection systems reaching a relatively high score of about 0.78 accuracy. However, since the number of claims (rumours) is relatively small, and the corpus is only based on tweets, this dataset alone is not suitable to train generally applicable claim validation systems.

Snopes17 A corpus featuring a substantially larger number of validated claims was introduced by Popat et al. (2017). It contains 4,956 claims annotated with verdicts which have been extracted from the Snopes website as well as the Wikipedia collections of proven hoaxes[10] and fictitious people[11]. For each claim, the authors extracted about 30 associated documents using the Google search engine, resulting in a collection of 136,085 documents. However, since the documents were not annotated by fact-checkers, irrelevant information is present and important information for the claim validation might be missing.

CLEF-2018 Another corpus concerned with political debates was introduced by Nakov et al. (2018) and used for the CLEF-2018 shared task. The corpus consists of transcripts of political debates in English and Arabic and provides annotations for two tasks: identification of check-worthy statements (claims) in the transcripts, and validation of 150 statements (claims) from the debates. However, as for the corpus **PolitiFact17**, no evidence for the validation of these claims is available.

FEVER18 The FEVER corpus introduced by Thorne et al. (2018a) is the largest available fact-

[9] http://www.emergent.info/

[10] https://en.wikipedia.org/wiki/List_of_hoaxes#Proven_hoaxe

[11] https://en.wikipedia.org/wiki/List_of_fictitious_people

checking corpus, consisting of 185,445 validated claims. The corpus is based on about 50k popular Wikipedia articles. Annotators modified sentences in these articles to create the claims and labeled other sentences in the articles, which support or refute the claim, as evidence. The corpus is large enough to train deep learning systems able to retrieve evidence from Wikipedia. Nevertheless, since the corpus only covers Wikipedia and the claims are created synthetically, the trained systems are unlikely to be able to extract evidence from heterogeneous web-sources and validate claims on the basis of evidence found on the Internet.

As our analysis shows, while multiple fact-checking corpora are already available, no single existing resource provides full fact-checking sub-task coverage backed by a substantially-sized and validated dataset spanning across multiple domains. To eliminate this gap, we have created a new corpus as detailed in the following sections.

3 Corpus construction

This section describes the original data from the Snopes platform, followed by a detailed report on our corpus annotation methodology.

3.1 Source data

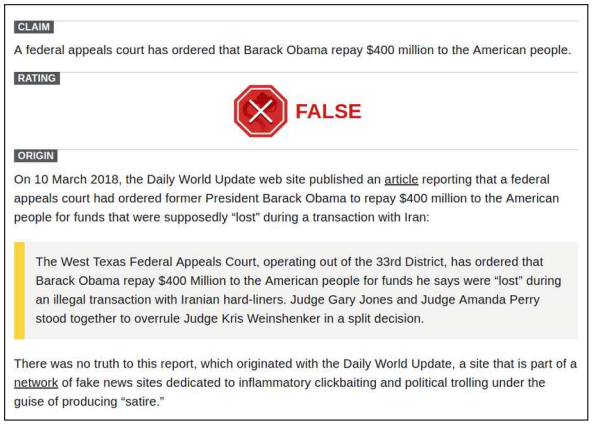


Figure 1: Snopes fact-checking data example

Snopes is a large-scale fact-checking platform that employs human fact-checkers to validate claims. A simple *fact-checking instance* from the Snopes website is shown in Figure 1. At the top of the page, the *claim* and the *verdict* (rating) are given. The fact-checkers additionally provide a *resolution* (origin), which backs up the verdict. Evidence in the resolution, which we call *evidence text snippets* (ETSs), is marked with a yellow bar. As additional validation support, Snopes fact-checkers provide URLs[12] for *original documents* (ODCs) from which the ETSs have been extracted or which provide additional information.

Our crawler extracts the *claims*, *verdicts*, *ETSs*, the *resolution*, as well as *ODCs* along with their URLs, thereby enriching the ETSs with useful contextual information. Snopes is almost entirely focused on claims made on English speaking websites. Our corpus therefore only features English fact-checking instances.

3.2 Corpus annotation

While ETSs express a stance towards the claim, which is useful information for the fact-checking process, this stance is not explicitly stated on the Snopes website. Moreover, the ETSs given by fact-checkers are quite coarse and often contain detailed background information that is not directly related to the claim and consequently not useful for its validation. In order to obtain an informative, high-quality collection of evidence, we asked crowd-workers to label the stance of ETSs and to extract sentence-level evidence from the ETSs that are directly relevant for the validation of the claim. We further refer to these sentences as *fine grained evidence* (FGE).

Stance annotation. We asked crowd workers on Amazon Mechanical Turk[13] to annotate whether an ETS *agrees* with the claim, *refutes* it, or has *no stance* towards the claim. An ETS was only considered to express a stance if it explicitly referred to the claim and either expressed support for it or refuted it. In all other cases, the ETS was considered as having *no stance*.

FGE annotation. We filtered out ETSs with *no stance*, as they do not contain supporting or refuting FGE. If an ETS was annotated as supporting the claim, the crowd workers selected only supporting sentences; if the ETS was annotated as refuting the claim, only refuting sentences were selected. Table 2 shows two examples of ETSs with annotated FGE. As can be observed, not all information given in the original ETS is directly relevant for validating the claim. For example, sentence (1c) in the first example's ETS simply provides additional background information and is therefore not considered FGE.

[12] underlined words in the resolution are hyperlinks

[13] https://www.mturk.com/

ETS stance: support
Claim: The Fox News will be shutting down for routine maintenance on 21 Jan. 2013.
Evidence text snippet: (1a) *Fox News Channel announced today that it would shutdown for what it called "routine maintenance".* (1b) *The shutdown is on 21 January 2013.* (1c) Fox News president Roger Ailes explained the timing of the shutdown: "We wanted to pick a time when nothing would be happening that our viewers want to see."

ETS stance: refute
Claim: Donald Trump supported Emmanuel Macron during the French election.
Evidence text snippet: (2a) In their first meeting, the U.S. President told Emmanuel Macron that he had been his favorite in the French presidential election saying "You were my guy". (2b) *In an interview with the Associated Press, however, Trump said he thinks Le Pen is stronger than Macron on what's been going on in France.*

Table 2: Examples of FGE annotation in supporting (top) and refuting (bottom) ETSs, sentences selected as FGE in italic.

4 Corpus analysis

4.1 Inter-annotator agreement

Stance annotation. Every ETS was annotated by at least six crowd workers. We evaluate the inter-annotator agreement between groups of workers as proposed by Habernal et al. (2017), i.e. by randomly dividing the workers into two equal groups and determining the aggregate annotation for each group using MACE (Hovy et al., 2013). The final inter-annotator agreement score is obtained by comparing the aggregate annotation of the two groups. Using this procedure, we obtain a Cohen's Kappa of $\kappa = 0.7$ (Cohen, 1968), indicating a substantial agreement between the crowd workers (Artstein and Poesio, 2008). The gold annotations of the ETS stances were computed with MACE, using the annotations of all crowd workers. We have further assessed the quality of the annotations performed by crowd workers by comparing them to expert annotations. Two experts labeled 200 ETSs, reaching the same agreement as the crowd workers, i.e. $\kappa = 0.7$. The agreement between the experts' annotations and the computed gold annotations from the crowd workers is also substantial, $\kappa = 0.683$.

FGE Annotation. Similar to the stance annotation, we used the approach of Habernal et al. (2017) to compute the agreement. The inter-annotator agreement between the crowd workers in this case is $\kappa = 0.55$ Cohen's Kappa. We compared the annotations of FGE in 200 ETSs by experts with the annotations by crowd workers, reaching an agreement of $\kappa = 0.56$. This is considered as *moderate* inter-annotator agreement (Artstein and Poesio, 2008).

In fact, the task is significantly more difficult than stance annotation as sentences may provide only partial evidence for or against the claim. In such cases, it is unclear how large the information overlap between sentence and claim should be for a sentence to be FGE. The sentence (1a) in Table 2, for example, only refers to one part of the claim without mentioning the time of the shutdown. We can further modify the example in order to make the problem more obvious: *(a) The channel announced today that it is planing a shutdown. (b) Fox News made an announcement today.*

As the example illustrates, there is a gradual transition between sentences that can be considered as essential for the validation of the claim and those which just provide minor negligible details or unrelated information. Nevertheless, even though the inter-annotator agreement for the annotation of FGE is lower than for the annotation of ETS stance, compared to other annotation problems (Zechner, 2002; Benikova et al., 2016; Tauchmann et al., 2018) that are similar to the annotation of FGE, our framework leads to a better agreement.

4.2 Corpus statistics

Table 3 displays the main statistics of the corpus. In the table, *FGE sets* denotes groups of FGE extracted from the same ETS. Many of the ETSs have been annotated as *no stance* (see Table 5) and, following our annotation study setup, are not used for FGE extraction. Therefore, the number of FGE sets is much lower than that of ETSs. We have found that, on average, an ETS consists of 6.5 sentences. For those ETS that have support/refute stance, on average, 2.3 sentences are selected as FGE. For many of the ETSs, no original documents (ODCs) have been provided (documents from which they have been extracted). On

the other hand, in many instances, links to ODCs are given that provide additional information, but from which no ETSs have been extracted.

entity:	claims	ETSs	FGE sets	ODCs
count:	6,422	16,509	8,291	14,296

Table 3: Overall statistics of the corpus

The distribution of verdicts in Table 4 shows that the dataset is unbalanced in favor of false claims. The label *other* refers to a collocation of verdicts that do not express a tendency towards declaring the claim as being false or true, such as *mixture, unproven, outdated, legend,* etc.

verdict:	false	true	most. false	most. true	other
count	2,943	659	334	93	2,393
%	45.8	10.3	5.2	1.4	37.3

Table 4: Distribution of verdicts for claims

Table 5 shows the stance distribution for ETSs. Here, supporting ETSs and ETSs that do not express any stance are dominating.

stance:	support	refute	no stance
ETSs:			
count	6,734	2,266	7,508
%	40.8	13.7	45.5
FGE sets:			
count	6,178	2,113	–
%	74.5	25.5	–

Table 5: Class distribution of ETSs the FGE sets

For supporting and refuting ETSs annotators identified FGE sets for 8,291 out of 8,998 ETSs. ETSs with a stance but without FGE sets often miss a clear connection to the claim, so the annotators did not annotate any sentences in these cases. The class distribution of the FGE sets in Table 5 shows that supporting ETSs are more dominant.

To identify potential biases in our new dataset, we investigated which topics are prevalent by grouping the fact-checking instances (claims with their resolutions) into categories defined by Snopes. According to our analysis, the four categories *Fake News*, *Political News*, *Politics* and *Fauxtography* are dominant in the corpus ranging from more than 700 to about 900 instances. A significant number of instances are present in the categories *Inboxer Rebellion (Email hoax)*, *Business*, *Medical*, *Entertainment* and *Crime*.

We further investigated the sources of the collected documents (ODCs) and grouped them into a number of classes. We found that 38% of the articles are from different news websites ranging from mainstream news like CNN to tabloid press and partisan news. The second largest group of documents are false news and satirical articles with 30%. Here, the majority of articles are from the two websites *thelastlineofdefense.org* and *worldnewsdailyreport.com*. The third class of documents, with a share of 11%, are from social media like Facebook and Twitter. The remaining 21% of documents come from diverse sources, such as debate blogs, governmental domains, online retail, or entertainment websites.

4.3 Discussion

I this subsection, we briefly discuss the differences of our corpus to the FEVER dataset as the most comprehensive dataset introduced so far. Due to the way the FEVER dataset was constructed, the claim validation problem defined by this corpus is different compared to the problem setting defined by our corpus. The verdict of a claim for FEVER depends on the stance of the evidence, that is, if the stance of the evidence is agree the claim is necessarily true, and if the stance is disagree the claim is necessarily false. As a result, the claim validation problem can be reduced to stance detection. Such a transformation is not possible for our corpus, as the evidence might originate from unreliable sources and a claim may have both supporting and refuting ETSs. The stance of ETSs is therefore not necessarily indicative of the veracity of the claim. In order to investigate how the stance is related to the verdict of the claim for our dataset, we computed their correlation. In the correlation analysis, we considered how a claims' verdict, represented by the classes *false*, *mostly false*, *other*, *mostly true*, *true*, correlates with the number of supporting ETSs minus the number of refuting ETSs. More precisely, the verdicts of the claims are considered as one variable, which can take 5 discreet values ranging from *false* to *true*, and the stance is considered as the other variable, which is represented by the difference between the number of supporting versus the number of refuting evidence. We found that the verdict is only weakly correlated with the stance, as indicated by the Pearson correlation coefficient of 0.16. This illustrates that the fact-checking problem setting

for our corpus is more challenging than for the FEVER dataset.

5 Experiments and error analysis

The annotation of the corpus described in the previous section provides supervision for different fact-checking sub-tasks. In this paper, we perform experiments for the following sub-tasks: (1) detection of the stance of the ETSs with respect to the claim, (2) identification of FGE in the ETSs, and (3) prediction of a claim's verdict given FGE.

There are a number of experiments beyond the scope of this paper, which are left for future work: (1) retrieval of the original documents (ODCs) given a claim, (2) identification of ETSs in ODCs, and (3) prediction of a claim's verdict on the basis of FGE, the stance of FGE, and their sources.

Moreover, in this paper, we consider the three tasks independent of each other rather than as a pipeline. In other words, we always take the gold standard from the preceding task instead of the output of the preceding model in the pipeline. For the three independent tasks, we use recently suggested models that achieved high performance in similar problem settings. In addition, we provide the human agreement bound, which is determined by comparing expert annotations for 200 ETSs to the gold standard derived from crowd worker annotations (Section 4.1).

5.1 Stance detection

In the stance detection task, models need to determine whether an ETS *supports* or *refutes* a claim, or expresses *no stance* with respect to the claim.

5.1.1 Models and Results

We report the performance of the following models: `AtheneMLP` is a feature-based multi-layer perceptron (Hanselowski et al., 2018a), which has reached the second rank in the Fake News Challenge. `DecompAttent` (Parikh et al., 2016) is a neural network with a relatively small number of parameters that uses decomposable attention, reaching good results on the Stanford Natural Language Inference task (Bowman et al., 2015). `USE+Attent` is a model which uses the Universal Sentence Encoder (USE) (Cer et al., 2018) to extract representations for the sentences of the ETSs and the claim. For the classification of the stance, an attention mechanism and a MLP is used.

The results in Table 6 show that `AtheneMLP` scores highest. Similar to the outcome of the Fake News Challenge, feature-based models outperform neural networks based on word embeddings (Hanselowski et al., 2018a). As the comparison to the human agreement bound suggests, there is still substantial room for improvement.

model	recall	precision	F1m
agreement bound	0.770	0.837	0.802
random baseline	0.333	0.333	0.333
majority vote	0.150	0.333	0.206
AtheneMLP	**0.585**	**0.607**	**0.596**
DecompAttent	0.510	0.560	0.534
USE+Attent	0.380	0.505	0.434

Table 6: Stance detection results (F1m = F1 macro)

5.1.2 Error analysis

We performed an error analysis for the best-scoring model `AtheneMLP`. The error analysis has shown that *supporting* ETSs are mostly classified correctly if there is a significant lexical overlap between the claim and the ETS. If the claim and the ETSs use different wording, or if the ETS implies the validity of the claim without explicitly referring to it, the model often misclassifies the snippets (see example in the Appendix A.2.1). This is not surprising, as the model is based on bag-of-words, topic models, and lexica.

Moreover, as the distribution of the classes in Table 5 shows, *support* and *no stance* are more dominant than the *refute* class. The model is therefore biased towards these classes and is less likely to predict *refute* (see confusion matrix in the Appendix Table 11). An analysis of the misclassified *refute* ETSs has shown that the contradiction is often expressed in difficult terms, which the model could not detect, e.g. "the myth originated", "no effect can be observed", "The short answer is no".

5.2 Evidence extraction

We define evidence extraction as the identification of fine-grained evidence (FGE) in the evidence text snippets (ETSs). The problem can be approached in two ways, either as a *classification problem*, where each sentence from the ETSs is classified as to whether it is an evidence for a given claim, or as a *ranking problem*, in the way defined in the FEVER shared task. For FEVER, sentences in introductory sections of Wikipedia articles need to be ranked according to their relevance for the validation of the claim and the 5 highest ranked sentences are taken as evidence.

5.2.1 Models and Results

We consider the task as a ranking problem, but also provide the human agreement bound, the random baseline and the majority vote for evidence extraction as a *classification problem* for future reference in Table 10 in the Appendix.

To evaluate the performance of the models in the *ranking* setup, we measure the precision and recall on five highest ranked ETS sentences (precision @5 and recall @5), similar to the evaluation procedure used in the FEVER shared task. Table 7 summarizes the performance of several models on our corpus. The `rankingESIM` (Hanselowski et al., 2018b) was the best performing model on the FEVER evidence extraction task. The `Tf-Idf` model (Thorne et al., 2018a) served as a baseline in the FEVER shared task. We also evaluate the performance of `DecompAttent` and a simple `BiLSTM` (Hochreiter and Schmidhuber, 1997) architecture. To adjust the latter two models to the ranking problem setting, we used the hinge loss objective function with negative sampling as implemented in the `rankingESIM` model. As in the FEVER shared task, we consider the recall @5 as a metric for the evaluation of the systems.

The results in Table 7 illustrate that, in terms of recall, the neural networks with a small number of parameters, `BiLSTM` and `DecompAttent`, perform best. The `Tf-Idf` model reaches best results in terms of precision. The `rankingESIM` reaches a relatively low score and is not able to beat the random baseline. We assume this is because the model has a large number of parameters and requires many training instances.

model	precision @5	recall @5
random baseline	0.296	0.529
BiLSTM	0.451	**0.637**
DecompAttent	0.420	0.627
Tf-Idf	**0.627**	0.601
rankingESIM	0.288	0.507

Table 7: Evidence extraction: ranking setting

5.2.2 Error analysis

We performed an error analysis for the `BiLSTM` and the `Tf-Idf` model, as they reach the highest recall and precision, respectively. `Tf-Idf` achieves the best precision because it only predicts a small set of sentences, which have lexical overlap with the claim. The model therefore misses FGE that paraphrase the claim. The `BiLSTM` is better able to capture the semantics of the sentences. We believe that it was therefore able to take related word pairs, such as "*Israel*" - "*Jewish*", "*price*"-"*sold*", "*pointed*"-"*pointing*", "*broken*"-"*injured*", into account during the ranking process. Nevertheless, the model fails when the relationship between the claim and the potential FGE is more elaborate, e.g. if the claim is not paraphrased, but reasons for it being true are provided. An example of a misclassified sentence is given in the Appendix A.2.2.

5.3 Claim validation

We formulate the claim validation problem in such a way that we can compare it to the FEVER *recognizing textual entailment* task. Thus, as illustrated in Table 8, we compress the different verdicts present on the Snopes webpage into three categories of the FEVER shared task. In order to form the *not enough information* (NEI) class, we compress the three verdicts *mixture*, *unproven*, and *undetermined*. We entirely omit all the other verdicts like *legend, outdated, miscaptioned*, as these cases are ambiguous and difficult to classify. For the classification of the claims, we provide only the FGE as they contain the most important information from ETSs.

FEVER	Snopes
refuted:	false, mostly false
supported:	true, mostly true
NEI:	mixture, unproven, undetermined

Table 8: Compression of Snopes verdicts

5.3.1 Experiments

For the claim validation, we consider models of different complexity: `BertEmb` is an MLP classifier which is based on BERT pre-trained embeddings (Devlin et al., 2018); `DecompAttent` was used in the FEVER shared task as baseline; `extendedESIM` is an extended version of the ESIM model (Hanselowski et al., 2018b) reaching the third rank in the FEVER shared task; `BiLSTM` is a simple BiLSTM architecture; `USE+MLP` is the Universal Sentence Encoder combined with a MLP; `SVM` is an SVM classifier based on bag-of-words, unigrams, and topic models.

The results illustrated in Table 9 show that `BertEmb`, `USE+MLP`, `BiLSTM`, and `extendedESIM` reach similar performance, with `BertEmb` being the best. However, compared to the FEVER claim validation problem,

Labeling method	recall m	prec. m	F1 m
random baseline	0.333	0.333	0.333
majority vote	0.198	0.170	0.249
BertEmb	0.477	**0.493**	**0.485**
USE+MLP	0.483	0.468	0.475
BiLSTM	0.456	0.473	0.464
extendedESIM	**0.561**	0.503	0.454
featureSVM	0.384	0.396	0.390
DecompAttent	0.336	0.312	0.324

Table 9: Claim validation results (m = macro)

where systems reach up to 0.7 F1 macro, the scores are relatively low. Thus, there is ample opportunity for improvement by future systems.

5.3.2 Error analysis

We performed an error analysis for the best-scoring model `BertEmb`. The class distribution for claim validation is highly biased towards *refuted* (false) claims and, therefore, claims are frequently labeled as *refuted* even though they belong to one of the other two classes (see confusion matrix in the Appendix in Table 12).

We have also found that it is often difficult to classify the claims as the provided FGE in many cases are contradicting (e.g. Appendix A.2.3). Although the corpus is biased towards false claims (Table 5), there is a large number of ETSs that support those false claims (Table 4). As discussed in Section 4.2, this is because many of the retrieved ETSs originate from false news websites.

Another possible reason for the lower performance is that our data is heterogeneous and, therefore, it is more challenging for a machine learning model to generalize. In fact, we have performed additional experiments in which we pre-trained a model on the FEVER corpus and fine-tuned the parameters on our corpus and vice versa. However, no significant performance gain could be observed in both experiments

Based on our analysis, we conclude that heterogeneous data and FGE from unreliable sources, as found in our corpus and in the real world, make it difficult to correctly classify the claims. Thus, in future experiments, not just FGE need to be taken into account, but also additional information from our newly constructed corpus, that is, the stance of the FGE, FGE sources, and documents from the Snopes website which provide additional information about the claim. Taking all this information into account would enable the system to find a consistent configuration of these labels and thus potentially help to improve performance. For instance, a claim that is *supported* by evidence coming from an *unreliable* source is most likely *false*. In fact, we believe that modeling the meta-information about the evidence and the claim more explicitly represents an important step in making progress in automated fact-checking.

6 Conclusion

In this paper, we have introduced a new richly annotated corpus for training machine learning models for the core tasks in the fact-checking process. The corpus is based on heterogeneous web sources, such as blogs, social media, and news, where most false claims originate. It includes validated claims along with related documents, evidence of two granularity levels, the sources of the evidence, and the stance of the evidence towards the claim. This allows training machine learning systems for document retrieval, stance detection, evidence extraction, and claim validation.

We have described the structure and statistics of the corpus, as well as our methodology for the annotation of evidence and the stance of the evidence. We have also presented experiments for stance detection, evidence extraction, and claim validation with models that achieve high performance in similar problem settings. In order to support the development of machine learning approaches that go beyond the presented models, we provided an error analysis for each of the three tasks, identifying difficulties with each.

Our analysis has shown that the fact-checking problem defined by our corpus is more difficult than for other datasets. Heterogeneous data and evidence from unreliable sources, as found in our corpus and in the real world, make it difficult to correctly classify the claims. We conclude that more elaborate approaches are required to achieve higher performance in this challenging setting.

7 Acknowledgements

This work has been supported by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) at the Technische Universität Darmstadt under grant No. GRK 1994/1.

References

Ron Artstein and Massimo Poesio. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Darina Benikova, Margot Mieskes, Christian M. Meyer, and Iryna Gurevych. 2016. Bridging the gap between extractive and abstractive summaries: Creation and evaluation of coherent extracts from heterogeneous sources. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 1039–1050, Osaka, Japan.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (System Demonstrations)*.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. Semeval-2017 task 8: Rumoureval: Determining rumour veracity and support for rumours. *Proceedings of the 11th International Workshop on Semantic Evaluations (SemEval-2017)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT)*.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data-set for stance classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT)*, pages 1163–1168, San Diego, CA, USA.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2017. The argument reasoning comprehension task. *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*.

Andreas Hanselowski and Iryna Gurevych. 2017. A framework for automated fact-checking for real-time validation of emerging claims on the web. *Proceedings of the NIPS Workshop on Prioritising Online Content (WPOC2017)*.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M Meyer, and Iryna Gurevych. 2018a. A retrospective analysis of the fake news challenge stance detection task. *Proceedings of the 2018 International Committee on Computational Linguistics*.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018b. Ukp-athene: Multi-sentence textual entailment for claim verification. *Proceedings of the EMNLP 2018 First Workshop on Fact Extraction and Verification*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning Whom to Trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL/HLT)*, pages 1120–1130, Atlanta, GA, USA.

Lee Howell et al. 2013. Digital wildfires in a hyperconnected world. *WEF Report*, 3:15–94.

Preslav Nakov, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Wajdi Zaghouani, Pepa Atanasova, Spas Kyuchukov, and Giovanni Da San Martino. 2018. Overview of the clef-2018 checkthat! lab on automatic identification and verification of political claims. In *Proceedings of the Ninth International Conference of the CLEF Association: Experimental IR Meets Multilinguality, Multimodality, and Interaction*, Lecture Notes in Computer Science, Avignon, France. Springer.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Dean Pomerleau and Delip Rao. 2017. The Fake News Challenge: Exploring how artificial intelligence technologies could be leveraged to combat fake news. http://www.fakenewschallenge.org/. Accessed: 2019-4-20.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 1003–1012. International World Wide Web Conferences Steering Committee.

Christopher Tauchmann, Thomas Arnold, Andreas Hanselowski, Christian M Meyer, and Margot Mieskes. 2018. Beyond generic summarization: A

multi-faceted hierarchical summarization corpus of large heterogeneous data. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018a. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018b. The fact extraction and verification (fever) shared task. *arXiv preprint arXiv:1811.10971*.

Andreas Vlachos and Sebastian Riedel. 2014. Fact checking: Task definition and dataset construction. In *Proceedings of the ACL 2014 Workshop on Language Technologies and Computational Social Science*, pages 18–22.

William Yang Wang. 2017. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Klaus Zechner. 2002. Automatic Summarization of Open-Domain Multiparty Dialogues in Diverse Genres. *Computational Linguistics*, 28(4):447–485.

.

Detecting Frames in News Headlines and Its Application to Analyzing News Framing Trends Surrounding U.S. Gun Violence

Siyi Liu **Lei Guo** **Kate Mays** **Margrit Betke** **Derry Tanti Wijaya**
Boston University
{liusiyi, guolei, kkmays, betke, wijaya}@bu.edu

Abstract

Different news articles about the same topic often offer a variety of perspectives: an article written about gun violence might emphasize gun control, while another might promote 2nd Amendment rights, and yet a third might focus on mental health issues. In communication research, these different perspectives are known as "frames", which, when used in news media will influence the opinion of their readers in multiple ways. In this paper, we present a method for effectively detecting frames in news headlines. Our training and performance evaluation is based on a new dataset of news headlines related to the issue of gun violence in the United States. This Gun Violence Frame Corpus (GVFC) was curated and annotated by journalism and communication experts. Our proposed approach sets a new state-of-the-art performance for multiclass news frame detection, significantly outperforming a recent baseline by 35.9% absolute difference in accuracy. We apply our frame detection approach in a large scale study of 88k news headlines about the coverage of gun violence in the U.S. between 2016 and 2018.

1 Introduction

The political climate in the United States is increasingly polarized (Pew Research Center, 2018a). To many media scholars and pundits, the main reason that liberals and conservatives inhabit different worlds is that news media of varied political orientations have been depicting two distinct versions of social reality (Mitchell et al., 2014; Stroud, 2011). To address this problem, one needs to assess the ways in which news reporters frame important public affairs. In communication research, "to frame" means "to select some aspects of a perceived reality and make them more salient in a communicating text" (Entman, 1993). Like any type of communication, news involves framing. In a polarized media environment, partisan media outlets intentionally frame news stories in a way to advance certain political agendas (Jamieson et al., 2007; Levendusky, 2013). Even when journalists make their best efforts to pursue objectivity, media framing often favors one side over another in political disputes, thus always resulting in some degree of bias (Entman, 2010). Hence, a news framing analysis is helpful because it not only tells us whether a news article is left- or right-leaning (or positive or negative), but also reveals how the article is structured to promote a certain side of the political spectrum.

In communication research, manual identification of media frames is a challenging task due to the large amount of media data in this news-saturated environment. More importantly, there is a high level of complexity in framing analysis that often requires a careful investigation of nuances in news coverage, which is time-consuming. In the field of Natural Language Processing (NLP), automated news framing analysis is a relatively unexplored area. Existing sentiment-analysis techniques fall short of addressing the nuances needed for framing analysis, which requires the detection of perspectives beyond positive and negative.

In this paper, we develop a neural network based approach for classifying frames in news article headlines by fine-tuning a state-of-the-art language representation model (BERT: Bidirectional Encoder Representations from Transformers (Devlin et al., 2018)) for the task of frame detection.

Here, we focus on the application of news frame detection on one prominent public affairs issue in the United States, namely, gun violence. Some of the deadliest mass shootings have happened during the past few years. In fact, the United States has the highest rate of gun-related homicides in the developed world. However, Republicans and Democrats remain divided on whether gun vio-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 504–514
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

lence is an important issue and disagree on most gun-related policies, making gun violence one of the most polarized issues in the country. (Pew Research Center, 2018b). As a result, despite the seriousness of the issue in reality, it is not considered a priority that should be tackled at the Congressional level. One factor that potentially explains the divergence of public opinion is how different politically oriented news media cover gun violence. It is likely that liberal-leaning and conservative-leaning media frame the issue in different ways, which may ultimately determine different publics' perception of the issue.

We use our frame detection approach to automatically detect frames of news article headlines related to gun violence during the past few years, which enables large scale analysis of framing trends surrounding this issue in the United States. Specifically, we focus on the years 2016, 2017, and 2018 because these three years have witnessed a number of high-profile mass shootings, which often reignited national gun debate.

Overall, our analysis results in interesting findings about U.S. media coverage of gun violence that speak to the divided media and political landscape in the country. Our contributions are twofold: Firstly, we have developed a state-of-the-art news frame detection approach by fine-tuning BERT language model to perform the multiclass (frame) classification on news article headlines. Our approach significantly outperforms a recent baseline in automated news frame detection (Field et al., 2018) and other neural network baselines.

Secondly, we have curated a new dataset of news articles related to U.S. gun violence: the Gun Violence Frame Corpus (GVFC), which contains news headlines and their frame annotations from 21 major U.S. news organizations. This dataset is the first of its kind in that it is carefully curated and contains domain-expert annotations of frames in news headlines. We use our model trained on GVFC to do a large scale analysis of U.S. gun violence framing trends in the U.S. between 2016 and 2018.

2 Related Work

2.1 News Framing

Framing is a subtle form of media manipulation in which some aspects of a topic are highlighted in order to promote a particular interpretation. It is related to the word choice and labeling by journalists (Hamborg et al., 2019) for example, by choosing "illegal alien" instead of "undocumented immigrant", journalists can highlight different aspects of an immigration issue.

Communication researchers have developed a variety of approaches to analyzing media framing. One popular quantitative approach is to first identify a list of frames and then manually classify news articles into one of the identified frames. Journalists often use generic frames that are common across a range of issues, such as human interest, conflict, and economic consequences (Russell Neuman et al., 1993; Nisbet, 2010; Semetko and Valkenburg, 2000), on top of issue-specific frames in their reporting. There are a number of issue-specific frames that have been particularly related to the issue of gun violence in the United States. On a basic level, the debate about guns has been framed as a threat to *public safety* (Haider-Markel and Joslyn, 2001; Lawrence and Birkland, 2004), enabled by weak *gun laws* (Birkland and Lawrence, 2009), versus an individual right to have access to guns secured by the *2nd Amendment*'s "right to bear arms" (Haider-Markel and Joslyn, 2001). Lawrence and Birkland (2004); Birkland and Lawrence (2009) also described how, after the Columbine shooting, the media discourse framed violent *popular culture* (e.g., movies and video games that glorify violence) as a culprit. Beyond the issue itself, the debate surrounding gun violence has also been framed as a Democrat vs. Republican *political contest* (Schnell, 2001).

In health communication, researchers have also examined the extent to which the news media frame the issue from the perspective of "dangerous people" (e.g., those with mental illness) wielding weapons as compared to "dangerous weapons" (e.g., large-capacity assault rifles) causing gun violence (McGinty et al., 2013). The *mental illness* of gunmen is often a focal point in the coverage of mass shootings (McGinty et al., 2014). Related to the issue of mental health are broader concerns about troubled individuals who lack the social support and resources to receive the help that they need (DeFoster and Swalve, 2018). The discussion about *race and ethnicity* has also emerged as a salient frame, in that news coverage of gun violence may differ somewhat depending on who the perpetrators are (Leavy and Maloney, 2009).

For our dataset, we detect these issue-specific frames typically found in media coverage of gun

violence, as well as generic frames like *economic consequences*. While it would be beneficial to automate the framing analysis across a variety of issues, here we argue that developing issue-specific tools will allow for a more nuanced understanding of each issue. Using our approach of combining expert-chosen frames for a particular issue and an automatic detection of these frames, communication researchers can further investigate how different news media influence public opinion in subtle ways *and* at scale, and thus be able to help prepare stronger arguments for journalistic practice and ultimately policy changes about the issue.

2.2 News Frame Detection

Media Frame Corpus (MFC) (Card et al., 2015) is one of the first large-scale datasets of frame annotations. It contains 11,900 hand-annotated English news articles for media framing that cover three issues: immigration, tobacco, and same-sex marriage. Undergraduate student annotators highlight the span of text that covers a frame following an annotation codebook. MFC has 15 generic media frames, which are defined in the *Policy Frames Codebook* by Boydstun et al. (2014), such as economics, political, quality of life, and also an "other" label for news articles that cannot be covered by any of the 15 frames. These news articles have been collected using keyword search from 13 national U.S. newspapers from 1990 to 2012 and contains 38,283 news articles. Duplicate and near-duplicate articles were removed and 20,037 of these articles were randomly selected for manual framing annotation. Aside from spans of text, headlines and entire news text are also annotated with the headline and primary frames respectively.

Naderi and Hirst (2017) detect news frames at the sentence-level using deep recurrent neural networks, specifically LSTM, BiLSTM, and GRU. They used news articles from MFC dataset (Card et al., 2015) to train and evaluate their model. They show that their results for frame detection are better than classifiers that rely on topics models for detecting frames (Tsur et al., 2015; Nguyen et al., 2015). Our work is different from theirs in that we focus on detecting the frame in the news article headline, which unlike a complete sentence, is typically a short phrase. We implement these deep recurrent networks in our experiments as baselines and find that our approach performs better for detecting frames in headlines, both in MFC and our GVFC. We also implement a recent word-based method for detecting frames in English and Russian news articles (Field et al., 2018) as another baseline. We detail these baseline approaches and their results in our experiment section (section 4).

3 Dataset Creation

3.1 News Article Collection

We drew our sample of news articles from a list of top U.S. news websites defined in terms of traffic to the websites. We cross-referenced several sources that had "top news sites" of their own: the Pew Research Center (2018b), Statista (2017), Alexa (2018), and MediaCloud, which is an open-source online platform. We synthesized these lists towards creating one list that contained news sites from the left, center, and right sides of the ideological spectrum based on categories defined in MediaCloud; Pew Research Center (2016); Ad Fontes Media (2019). We started with list of 30 media outlets based on these references.

We collected articles from these outlets from four time periods over the course of 2018 in order to capture a diversity of articles. Some articles were collected over periods during or immediately after a mass shooting (e.g., the Parkland School shooting in 02/2018). Other articles were collected when gun violence was not necessarily the most salient current event. We also included articles from several months before the 2018 U.S. midterm elections as the gun-related issue was a central topic for political discussion during this period. The articles were retrieved using Crimson Hexagon's ForSight social media analytics platform (Hexagon, 2018), retrieving articles that had at least one keyword in their headlines from the following list: {"gun," "firearm," "NRA," "2nd amendment," "second amendment," "AR15," "assault weapon," "rifle," "Brady act," "Brady bill," "mass shooting"}. We came up with the list of keywords based on the previous literature and on the review of a sample of our data. After collecting the articles, news articles with duplicate titles were removed and the rest sampled to be analyzed and annotated. After sampling and annotation, the final dataset contains frame annotations of news articles from a total of 21 media outlets[1].

[1]For reproducibility and future research, we make our dataset and annotation codebook publicly available at https://derrywijaya.github.io/GVFC.html

3.2 News Article Annotation

Quantitative content analysis (QCA) in communication research is a commonly used method to derive "replicable and meaningful inferences from texts (or other meaningful matter)" (Krippendorff, 2004). To perform QCA, one draws a representative sample of text (or other types of content), on which two or more trained *coders* (i.e., annotators) apply a codebook protocol, which should include all of the variables for annotation and their definitions. Prior to coding the entire sample independently, coders are first trained on the codebook, and their agreement on how to apply the codes is measured with inter-coder reliability (ICR). High ICR values implies that two or more coders consistently categorized the content similarly, which signals a high validity of the coded results. Once coders have reached an acceptable ICR (above 90% agreement or 0.70 Krippendorff α (Krippendorff, 2004)), they can code the rest of the sample independently.

Codebook Creation Our codebook was developed by drawing from the literature on framing gun violence, described earlier, as well as from a preliminary analysis of the data. This resulted in 9 frames, including both generic: "Politics", "Public opinion", "Society/Culture", and "Economic consequences" and issue-specific: "2nd Amendment" (Gun Rights), "Gun control/regulation", "Mental health", "School/Public space safety", and "Race/Ethnicity".

Unit of Annotation We choose our unit of annotation to be a news headline for several reasons. Firstly, psychologists have long argued that first impressions are lasting impressions (Digirolamo and Hintzman, 1997). This thesis applies to news reading behavior as well. Media framing researchers often identify and measure frames in news headlines (e.g., (Bleich et al., 2015; Trimble and Sampert, 2004), which are seen by the audience first and can determine the perception of the text that follows (Tankard Jr, 2001). As Pan and Kosicki (1993) suggests, a headline is "the most salient cue to activate certain semantically related concepts in readers minds; it is thus the most powerful framing device of the syntactical structure".

Secondly, the analysis of news headlines became more relevant in the emerging (i.e. digital) media environment where a large portion of people only read headlines but nothing else (Gabielkov et al., 2016). Further, driven by the attention economy, many online media even use news headlines as "clickbait", presenting sensational but misleading information that deviates from the content included in the actual news story (Chen et al., 2015). That is, a news story may be framed differently in its headline and the rest of the article. In cases like this, research shows that even reading through the article cannot necessarily correct the headlines misdirection (Ecker et al., 2014). Taken together, detecting frames through news headlines provides the most direct clue to the potential influence of the news coverage.

Annotation Process Two communication graduate students were recruited to annotate a sample of the collected news articles. They were instructed to first determine whether the news headline was relevant to gun violence in the United States. If yes, they were asked to identify up to two dominant frames in the headline. They were trained on the codebook during the training sessions. In the first training session, the students were given a 100-headline sample to code, and ICR was not met. Hence, a second training session was held to further clarify the codebook and resolve any confusion. The students coded another 100-headline sample, for which ICR was met on all variables: relevance (99% agreement, 0.97 α), frame A (94.10% agreement, 0.90 α) and frame B (96.04% agreement, 0.82 α). Following QCA, once the ICR was met, one student continued to code another 2,790 news headlines, resulting in a total of 2,990 annotated news headlines.

3.3 Dataset Properties

GVFC includes 2,990 news headlines, 1,300 of which are annotated as relevant to the gun violence issue in the United States. Out of the relevant headlines, only 319 are found to have 2 frames. Examples of headlines with 2 frames are "It's Time to Hand the Mic to Gun Owners", annotated with "Public opinion" (frame A) and "2nd Amendment" (frame B); and "Trevor Noah: 'The Second Amendment Is Not Intended for Black People", annotated with "2nd Amendment" (frame A) and "Race/Ethnicity" (frame B).

We use frame A annotations to train our frame classification model but find that our model also identifies some of frame B annotations in its top predictions (Section 4.2). Table 1 shows frame A class distribution in GVFC that reflects the varying

coverage of different frames in the U.S. media.

4 Experiments

We use the most recent method for automatic news frame detection (Field et al., 2018) as one of our baselines. They devise a word-based method for detecting the frames in English and Russian news. They use MFC to derive a lexicon for each frame F by computing pointwise mutual information $I(F, w)$ (Church and Hanks, 1990) for each word w and each frame F in the corpus. Each frame F's lexicon contains the top 250 words with the highest $I(F, w)$ for frame F. A news article has a frame F if it contains at least 3 instances of a word from F's lexicon with the primary frame being the most common frame, based on the number of words from each frame's lexicon in the document. We create lexicons for the 9 frames in our GVFC dataset and use them to compute the primary frames of news headlines.

We also implement LSTM-based neural networks for a more comprehensive evaluation. Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture that is widely used today in text classification tasks. There are plenty of variants from this type of architecture: Gated Recurrent Unit (GRU), Bi-directional LSTM, and Bi-directional GRU. We implement these networks with attention mechanism (Bahdanau et al., 2015) and use 100-dimensional pre-trained Glove embeddings (Pennington et al., 2014) as our initial word representations. We train and evaluate these networks for headlines frame classification with 128 units of RNN cells and one layer of attention mechanism at the end, a batch size of 128 for 2000 steps. We use Adam optimizer with a learning rate of 0.01.

As the results in Table 1 show, Bi-directional GRU with attention achieves the highest accuracy among our baselines. The reason behind this could be the fact that we have a small dataset and GRU needs fewer data points to generalize (Kaiser and Sutskever, 2015; Yin et al., 2017). Furthermore, the attention mechanism and bi-directionality allows for more contextual interpretation of the headlines and better detection of their frames.

4.1 News Frame Detection with BERT

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) take this idea of attention and bi-directionality further by building on the Transformer's encoder model that solely relies on multi-layer self-attention to compute contextual representations of its input, dispensing with any kinds of recurrence (Vaswani et al., 2017). The encoder is composed of a stack of identical layers, where each layer contains a self-attention mechanism, which allows the encoder to look at other words in the input sentence as it encodes the contextual representation of each word in the sentence, and a fully connected feed-forward network. The self-attention mechanism computes three vectors from the embedding of each word in the input sentence: the query q, key k, and value v vectors. It then computes the contextual representation of each word w in the sentence as the weighted sum of the value vectors of all the words in the sentence, where the weights are the scaled then normalized dot products between w's query vector and the key vectors of all the words in the sentence. The weights essentially determine how much focus to place on other parts of the input sentence as the encoder encodes a word at a certain position. Given that the query, key, and value vectors are computed by multiplying the input word embeddings matrix X with weight matrices learned during training, W^Q, W^K, W^V, the self-attention output can be formulated in matrix form as: $\text{Attention}(Q, K, V) = \text{softmax}(QK^T/\sqrt{d_k})V$ where $Q = XW^Q$, $K = XW^K$, $V = XW^V$.

BERT's encoder implements the Transformer's multi-layer self-attention mechanisms and fully utilizes its strength in storing the left and right context of each token by using a "masked language model" (MLM) pre-training objective, inspired by the Cloze task (Taylor, 1953). In its pre-training, BERT randomly masks some of the tokens from its input, and predicts the original vocabulary id of the masked word based only on its context. Unlike left-to-right language model pre-training, the MLM objective enables the representation to fuse the left and the right context, which allows BERT to pre-train a deep bidirectional Transformer representations from unlabeled large text corpora.

We fine-tune the pre-trained BERT-based uncased model on our multiclass frame classification by adding a frame classification layer on top of the model and fine-tune all the parameters end-to-end. Given a headline, BERT tokenizes the headline to tokens based on WordPiece tokens (Wu et al., 2016) and appends a special classification token

Frame Class	# Headlines	Baseline	LSTM	Bi-LSTM	Bi-LSTM w/ Attention	Bi-GRU w/ Attention	BERT
2nd Amendment	38	44.74	23.68	21.05	44.74	26.32	65.79
Gun control/regulation	215	50.23	63.72	66.51	72.09	76.28	84.19
Politics	373	40.48	78.28	77.75	84.18	85.79	89.54
Mental health	65	35.38	50.77	40.00	58.46	60.00	78.46
School/Public space safety	137	39.42	48.91	50.36	54.74	58.39	78.10
Race/Ethnicity	114	67.54	75.44	71.93	84.21	81.28	92.11
Public opinion	237	63.29	70.46	72.15	75.53	77.22	86.08
Society/Culture	41	43.90	24.39	19.51	36.59	21.95	58.54
Economic consequences	80	38.75	45.00	51.25	61.25	60.00	80.00
Overall	1300	48.38	64.37	64.48	72.15	72.76	84.23

Table 1: Class distribution of frame A annotations and micro-accuracies for the baseline (Field et al., 2018), LSTM, bi-directional LSTM, bi-directional LSTM and bi-directional GRU with attention, and our method based on fine tuning BERT.

([CLS]) at the beginning of the headline. We use the final hidden vector $C \in \mathbb{R}^H$ corresponding to [CLS] as the aggregate representation of the headline that is input to the classification layer (since encoding this token with self-attention effectively includes attention to all the tokens in the headline). The only new parameters are our classification layer weights $W \in \mathbb{R}^{K \text{x} H}$, where K = 9, the number of our frame classes. Given the imbalance in our class distribution, we compute a focal loss (FL) (Lin et al., 2017) that improves our classification performance compared to the standard cross entropy loss. We compute $\text{FL}(p) = -\alpha(1-p)^{\gamma}\log(p)$, where $p \in \mathbb{R}^K$ contains the probabilities of classifying the headline into each of the K frames i.e., $p = \text{softmax}(CW^T)$ and $\alpha \in \mathbb{R}^K$ contains the weighting factors, which we set for each frame to be its normalized inverse class frequency $\in [0, 1]$: the smaller the class, the higher the α and vice versa, which balances the importance of each class' examples. The modulating factor: $(1-p)^{\gamma}$ in FL down-weights the loss contribution of the easy examples – those that are well classified (i.e. have high p_k) – and thus focuses the training on hard-to-classify examples. Following Lin et al. (2017), we use $\gamma = 2$.

We train for 10 epochs with a batch size of 4, 2e-5 learning rate, and maximum sequence length of 128 tokens. Training and testing on the same stratified folds that we use for all our baselines, we achieve a 5-fold cross validation micro-accuracy of 84.23%. Our method based on BERT significantly outperforms not only the most recent news frame classification baseline, but also some state-of-the-art deep classification models, including bi-directional LSTM/GRU with attention on every frame of our GVFC dataset (Table 1).

We also evaluate our method to classify frames of news headlines in another dataset (MFC). As

MFC Issue	# Headlines	Bi-GRU w/ Attention	BERT
Immigration (I)	7231	40.84	52.38
Tobacco (T)	3959	57.20	67.94
Samesex (SS)	3842	61.57	71.50
I (top-5 frames)	4175	53.65	67.28
T (top-5 frames)	2759	71.44	82.32
SS (top-5 frames)	2937	74.94	83.07

Table 2: 10-fold cross-validation micro-accuracy on the MFC dataset for our best baseline from previous evaluation, and our model based on BERT.

Table 2 shows, our method significantly outperforms our top-performing baseline, both on the 15-frame classification task and on the top-5 (most frequent) frame classification on all issues: immigration, tobacco, and same-sex marriage. This shows that our method can perform well for detecting frames in headlines in different datasets and across a diverse range of issues.

4.2 Discussion

Our results show that fine-tuning on BERT performs well even on a small dataset like GVFC, which agrees with the findings of Devlin et al. (2018) that fine-tuning on BERT's pre-trained model can lead to large improvements even on very small scale tasks. Part of the reasons may be due to BERT's deep attention mechanism. Attention mechanism has been shown to be data-efficient and helps improve performance significantly even when the dataset is small (Vinyals et al., 2015). Even adding standard attention improves the accuracy of our LSTM-based baselines significantly (Table 1). BERT's success can also be traced to its design of bidirectional Transformer that offers richer contextual information. Furthermore, BERT was pre-trained on a large corpus to produce this representation. Fine-tuning on BERT allows us to transfer this contextual knowledge to

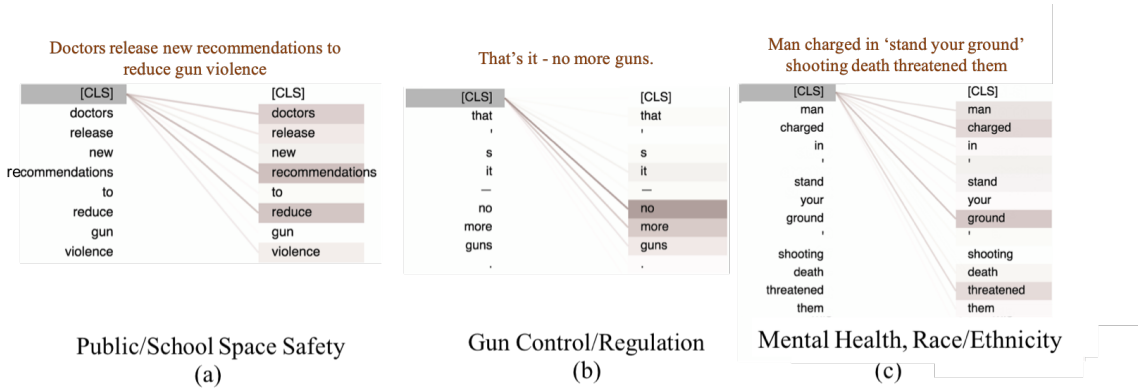


Figure 1: Visualization of our fine-tuned model, the headlines and the predicted frames. The thicker the line, the more attention placed on the token for computing the aggregate i.e., [CLS] representation for the classification

classifying frames in headlines, which are very short compared to the entire news text. The ability to transfer contextual knowledge from a large corpus leads to better representation for these short pieces of texts and better generalization of our model compared to the lexicon-baseline that only relies on word-frame co-occurrences in GVFC.

We use a visualization tool (Vig, 2019) to obtain insights into *what* our fine-tuned model is attending to when making decisions. For example, we observe that pre-training on a large corpus may have helped our model predict the frame "School/Public Space Safety" for the headline: "Doctors release new recommendations to reduce gun violence" by attending to words like "Doctors" and "recommendations" (Figure 1(a)). Although these words do not co-occur frequently with this specific frame in GVFC, they may be related to school/public safety in general. The lexicon model, on the other hand, incorrectly predicts the "Gun control/regulation" frame due to the words "release" and "gun" in the headline.

Because news framing is closely related to journalists' word choice (Hamborg et al., 2019), we find that on frames such as "Race/Ethnicity," which has a specific set of keywords that the model can attend to like "black", "white", or "anti-semitic", both our model and the lexicon-baseline perform the best on this frame.

On the other hand, the performance of our model and the baseline differ significantly for generic frames such as "Politics," whose keywords may overlap with issue-specific frames such as "Gun control/regulation". Since BERT is pre-trained to take context into consideration, words like "gun", which appears with all the frames, can have different contextual representation depending on its context i.e., "gun lobby" vs. "gun permit". For example, the headline "That's it – no more guns" is classified correctly by BERT as having "Gun control/regulation" frame by attending to the context "no more" of "guns" (Figure 1(b)).

Also, despite not being trained to predict multiple frames, some of BERT's predictions of what it believes to be top frames align with that of human experts. There are 319 headlines in GVFC that were annotated with two frames: frame A and B, meaning that the headline is equally likely to belong to either frame. In our experiments, we only train our model with frame A annotations. However, we notice that out of the 319 headlines that have two frames, 164 of them have both frames predicted in the top-2 predictions of our model, showing the potential to fine-tune BERT for multi-label multi-class frame classification, which we will explore in the future. Furthermore, the accuracy of our model on GVFC increases to 87.92% if we consider our model's prediction to be correct if it predicts either frame for these 319 headlines.

More interestingly, we observe that our model can predict additional frames that may be applicable to the headlines but are not annotated. In Figure 1(c) for example, for the headline "Man charged in 'stand your ground' shooting death threatened them", our model first attends to the word "ground" and then "threatened" and predicts the frame "Race/Ethnicity" and then "Mental health". Even though this headline was only annotated with the "Mental health" frame (possibly due to the word "threatened" which, in the "Mental Health" description of the annotation codebook, may be referring to an individual's behaviors that indicate instability, impulsivity, anger, etc.), we believe that in this particular headline the

"Race/Ethnicity" frame is more applicable given the presence of 'stand your ground', a legislation that has been shown to have a quantifiable racial bias (Ackermann et al., 2015).

Overall, what we observe from visualizing our model suggests that the model is able to generalize beyond word-frame co-occurrences in the limited annotations by virtue of the contextual knowledge transfer obtained by pre-training on a large corpus.

5 Framing Trends, Analysis, and Conclusion

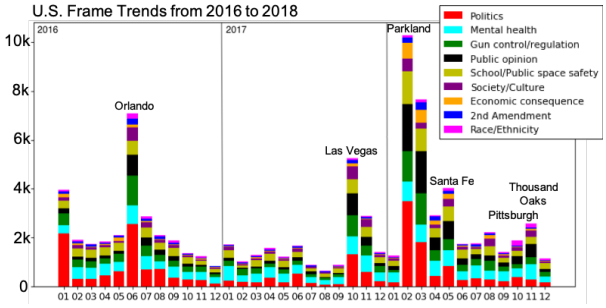


Figure 2: The number of times each frame is represented in new article headlines related to gun violence across the 3-year (per month) period. Some of the peaks represent the deadliest mass shootings in the U.S. since 1949 (CNN Library, 2019).

We used the same search words to retrieve news article headlines from the 21 U.S.news media outlets from 2016 to 2018. To apply our framing analysis, we first train a model to predict whether a news headline is relevant to the issue of U.S. gun violence by fine-tuning BERT-base uncased using the relevance annotations in GVFC. This relevance prediction model achieves a 10-fold cross validation precision of 0.93, 0.95 recall, and 0.94 F-score. We apply this model to find relevant headlines among the 88,470 collected, and apply our frame classification on the relevant headlines.

Several patterns emerged from the framing analysis. It appears that news media of all types have largely politicized the gun violence issue right after each major mass shooting (Figure 2). The focus on party politics, the divide between Democrats and Republicans on the issue dominated the coverage. This finding speaks to the highly polarized political environment in the U.S.

We also observe in Figure 2 that right after the Parkland school shooting in 02/2018, the discussion surrounding "Public opinion", "School/Public space safety", and "Economic consequences" frames increases. The increase in "Public opinion" and "School/Public space safety" frames is due to the growth of student activism in the wake of the shooting. Meanwhile, the increase in the "Economic Consequence" frame is due to the decision of several major companies such as Dick's Sporting Goods to stop selling assault-style weapons in the wake of the event.

We also observe in Figure 2 that frames that spike during every major shooting event, such as "Politics"/"Public opinion", are not the most persistent. Their frequency peaked during the month but dropped, often drastically, after. Notably, the "Mental health" frame (the cyan bar) appears to be the most persistent, appearing consistently over time in coverage about gun violence.

Another noticeable cross-media pattern in the U.S. media coverage of the gun violence issue is that the conservative-leaning and neutral media emphasized the mental health of individual gunmen to a greater extent than liberal-leaning media (see the cyan bar representing the "Mental health" frame in the left, center, and right plots of Figure 3). About a quarter of news articles from neutral and conservative-leaning media in 2017 are classified as having the "Mental Health" frame (27% and 22% of the articles respectively). In comparison, only 8% of news articles from liberal-leaning media are classified as having this "Mental Health" frame.

This finding about the conservative media (Figure 3 right) is not surprising because connecting mental illness and mass shooting has been a common stance among pro-gun Republican leaders (i.e., "guns don't kill people, people kill people"). More surprisingly though (and contrary to the common perception of mainstream media such as NYT, CNN, and CBS being liberal-leaning), our study suggests that these neutral, mainstream media (Figure 3 center) has also largely framed the issue from the aspect of mental health, often *more* than the conservative media, which may indicate conservative media's strong agenda-setting power in the U.S. media ecosystem.

Media framing scholars have also pointed out the importance of examining what aspects of the story has been left out. In our analysis, the frame of "Society/Culture" – a frame that is important and yet would not attract much web traffic – has not been a focus of gun violence coverage in the U.S. As the results demonstrate (Figure 3), major

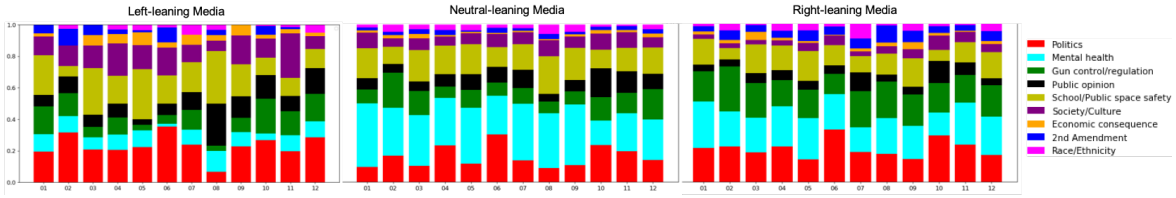


Figure 3: The proportion of each frame occurring in the 2017 news from news sites of different political leanings.

shootings were only able to trigger liberal-leaning media to pay more attention to this frame. 16% of articles from liberal-leaning media in 2017 are classified as having the "Society/Culture" frame, while only 9% of articles from neutral media (and only 5% from conservative-leaning media) are classified as having this "Society/Culture" frame. The lack of framing focus on the underlying cultural/societal issues as well as the aforementioned focus on party politics and strong agenda-setting speak to the status quo of the current U.S. news environment: profit-driven, sensational, and highly partisan.

In conclusion, we have presented in this paper a method for news headline frame classification that achieves state-of-the-art performance. We also release the codebook and a carefully curated Gun Violence Frame Corpus (GVFC) news articles whose headlines have been annotated with their corresponding frames by domain experts. We demonstrate the application of our framing detection to analyze a large corpus of news headlines for framing trends surrounding the U.S. gun violence coverage. We observe interesting findings and believe that frame detection and analysis can potentially be used to gain a deeper understanding of various issues of public affairs. Automatically detected frames in news headlines can also be used to curate more balanced news collections on various issues and perspectives.

Acknowledgement We thank the anonymous reviewers for their insightful comments. We thank Mona Jalal and Sha Lai for their help in data collection and figure plotting. This work is supported in part by the U.S. NSF grant 1838193. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsor.

References

Nicole Ackermann, Melody S Goodman, Keon Gilbert, Cassandra Arroyo-Johnson, and Marcello Pagano. 2015. Race, law, and health: Examination of stand your groundand defendant convictions in florida. *Social Science & Medicine*, 142:194–201.

Ad Fontes Media. 2019. The media bias chart, Last accessed on May 31, 2019. https://www.adfontesmedia.com/intro-to-the-media-bias-chart/.

Alexa. 2018. The top 500 sites on the web – subcategory "Breaking News." retrieved from https://www.alexa.com/topsites/category/top/news/breaking_news Last accessed on November 17, 2018.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.

Thomas A Birkland and Regina G Lawrence. 2009. Media framing and policy change after Columbine. *American Behavioral Scientist*, 52(10):1405–1425.

Erik Bleich, Hannah Stonebraker, Hasher Nisar, and Rana Abdelhamid. 2015. Media portrayals of minorities: Muslims in british newspaper headlines, 2001–2012. *Journal of Ethnic and Migration Studies*, 41(6):942–962.

Amber E. Boydstun, Dallas Card, Justin Gross, Paul Resnick, and Noah A. Smith. 2014. Tracking the development of media frames within and across policy issues. Technical report, University of California, Davis.

Dallas Card, Amber E Boydstun, Justin H Gross, Philip Resnik, and Noah A Smith. 2015. The media frames corpus: Annotations of frames across issues. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 438–444.

Yimin Chen, Niall J Conroy, and Victoria L Rubin. 2015. Misleading online content: Recognizing clickbait as false news. In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*, pages 15–19. ACM.

Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

CNN Library. 2019. Deadliest mass shootings in modern us history fast facts. https://edition.cnn.com/2013/09/16/us/20-deadliest-mass-shootings-in-u-s-history-fast-facts/index.html Last accessed on May 28th, 2019.

Ruth DeFoster and Natashia Swalve. 2018. Guns, culture or mental health? Framing mass shootings as a public health crisis. *Health communication*, 33(10):1211–1222.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Gregory J Digirolamo and Douglas L Hintzman. 1997. First impressions are lasting impressions: A primacy effect in memory for repetitions. *Psychonomic Bulletin & Review*, 4(1):121–124.

Ullrich KH Ecker, Stephan Lewandowsky, Ee Pin Chang, and Rekha Pillai. 2014. The effects of subtle misinformation in news headlines. *Journal of experimental psychology: applied*, 20(4):323.

Robert M Entman. 1993. Framing: Toward clarification of a fractured paradigm. *Journal of communication*, 43(4):51–58.

Robert M Entman. 2010. Media framing biases and political power: Explaining slant in news of campaign 2008. *Journalism*, 11(4):389–408.

Anjalie Field, Doron Kliger, Shuly Wintner, Jennifer Pan, Dan Jurafsky, and Yulia Tsvetkov. 2018. Framing and agenda-setting in Russian news: A computational analysis of intricate political strategies. *arXiv preprint arXiv:1808.09386*.

Maksym Gabielkov, Arthi Ramachandran, Augustin Chaintreau, and Arnaud Legout. 2016. Social clicks: What and who gets read on twitter? *ACM SIGMETRICS Performance Evaluation Review*, 44(1):179–192.

Donald P Haider-Markel and Mark R Joslyn. 2001. Gun policy, opinion, tragedy, and blame attribution: The conditional influence of issue frames. *Journal of Politics*, 63(2):520–543.

Felix Hamborg, Anastasia Zhukova, and Bela Gipp. 2019. Illegal aliens or undocumented immigrants? Towards the automated identification of bias by word choice and labeling. Technical report, University of Konstanz, Germany.

Crimson Hexagon. 2018. ForSight social media analytics platform, Last accessed on November 1, 2018.

Kathleen Hall Jamieson, Bruce Hardy, and Daniel Romer. 2007. The effectiveness of the press in serving the needs of American democracy. *A republic divided*, pages 21–51.

Łukasz Kaiser and Ilya Sutskever. 2015. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*.

Klaus Krippendorff. 2004. Content analysis: An introduction to its methodology.

Regina G Lawrence and Thomas A Birkland. 2004. Guns, Hollywood, and school safety: Defining the school-shooting problem across public arenas. *Social Science Quarterly*, 85(5):1193–1207.

Patricia Leavy and Kathryn P Maloney. 2009. American reporting of school violence and people like us: A comparison of newspaper coverage of the Columbine and Red Lake school shootings. *Critical Sociology*, 35(2):273–292.

Matthew S Levendusky. 2013. Why do partisan media polarize viewers? *American Journal of Political Science*, 57(3):611–623.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Emma E McGinty, Daniel W Webster, and Colleen L Barry. 2013. Effects of news media messages about mass shootings on attitudes toward persons with serious mental illness and public support for gun control policies. *American Journal of Psychiatry*, 170(5):494–501.

Emma E McGinty, Daniel W Webster, Marian Jarlenski, and Colleen L Barry. 2014. News media framing of serious mental illness and gun violence in the United States, 1997-2012. *American Journal of Public Health*, 104(3):406–413.

MediaCloud. 2018. https://mediacloud.org. Last accessed on November 15, 2018.

Amy Mitchell, Jeffrey Gottfried, Jocelyn Kiley, and Katerina Eva Matsa. 2014. Political polarization & media habits. *Pew Research Center*, 21.

Nona Naderi and Graeme Hirst. 2017. Classifying frames at the sentence level in news articles. *Policy*, 9:4–233.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, and Kristina Miler. 2015. Tea party in the house: A hierarchical ideal point topic model and its application to republican legislators in the 112th congress. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1438–1448.

Matthew C Nisbet. 2010. Knowledge into action: Framing the debates over climate change and poverty. In *Doing news framing analysis*, pages 59–99. Routledge.

Zhongdang Pan and Gerald M Kosicki. 1993. Framing analysis: An approach to news discourse. *Political communication*, 10(1):55–75.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pew Research Center. 2016. Ideological placement of each sources audience, Last accessed on May 31, 2019. https://www.pewresearch.org/pj$_1$4 − 10 − $21_m ediapolarization - 08 - 2/$.

Pew Research Center. 2018a. The partisan divide on political values grows even wider, Last accessed on November 17, 2018. http://www.people-press.org/2017/10/05/the-partisan-divide-on-political-values-grows-even-wider.

Pew Research Center. 2018b. State of the news media methodology, Last accessed on November 17, 2018. http://www.journalism.org/2018/07/25/state-of-the-news-media-methodology.

W Russell Neuman, Marion Just, and Ann Crigler. 1993. Common knowledge: News and the construction of political meaning. *Bibliovault OAI Repository, the University of Chicago Press*, 108.

Frauke Schnell, Karen Callaghan. 2001. Assessing the democratic debate: How the news media frame elite policy discourse. *Political communication*, 18(2):183–213.

Holli A Semetko and Patti M Valkenburg. 2000. Framing European politics: A content analysis of press and television news. *Journal of communication*, 50(2):93–109.

Statista. 2017. Most popular news websites as of August 2019, by unique monthly visitors. https://www.statista.com/statistics/381569/leading-news-and-media-sites-usa-by-share-of-visits/ Last accessed on November 17, 2018.

Natalie Jomini Stroud. 2011. *Niche news: The politics of news choice*. Oxford University Press on Demand.

James W Tankard Jr. 2001. The empirical approach to the study of media framing. In *Framing public life*, pages 111–121. Routledge.

Wilson L Taylor. 1953. cloze procedure: A new tool for measuring readability. *Journalism Bulletin*, 30(4):415–433.

Linda Trimble and Shannon Sampert. 2004. Who's in the game? the framing of election 2000 by the globe and mail and the national post. *Canadian Journal of Political Science/Revue canadienne de science politique*, 37(1):51–71.

Oren Tsur, Dan Calacci, and David Lazer. 2015. A frame of mind: Using statistical models for detection of framing and agenda setting campaigns. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1629–1638.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jesse Vig. 2019. Visualizing attention in transformer-based language models. *arXiv preprint arXiv:1904.02679*.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Petrov Slav, Sutskever Ilya, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proc. of NIPS*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

Learning A Unified Named Entity Tagger From Multiple Partially Annotated Corpora For Efficient Adaptation

Xiao Huang[1,2], **Li Dong**[3]*, **Elizabeth Boschee**[1], **Nanyun Peng**[1,2]
[1]Information Sciences Institute, University of Southern California
[2]Department of Computer Science, University of Southern California
[3]Outreach, Inc.
{huan183, lidong}@usc.edu, {boschee, npeng}@isi.edu

Abstract

Named entity recognition (NER) identifies typed entity mentions in raw text. While the task is well-established, there is no universally used tagset: often, datasets are annotated for use in downstream applications and accordingly only cover a small set of entity types relevant to a particular task. For instance, in the biomedical domain, one corpus might annotate genes, another chemicals, and another diseases—despite the texts in each corpus containing references to all three types of entities. In this paper, we propose a deep structured model to integrate these "partially annotated" datasets to *jointly* identify all entity types appearing in the training corpora. By leveraging multiple datasets, the model can learn robust input representations; by building a joint structured model, it avoids potential conflicts caused by combining several models' predictions at test time. Experiments show that the proposed model significantly outperforms strong multi-task learning baselines when training on multiple, partially annotated datasets and testing on datasets that contain tags from more than one of the training corpora.[1]

1 Introduction

Named Entity Recognition (NER), which identifies the boundaries and types of entity mentions from raw text, is a fundamental problem in natural language processing (NLP). It is a basic component for many downstream tasks, such as relation extraction (Hasegawa et al., 2004; Mooney and Bunescu, 2005), coreference resolution (Soon et al., 2001), and knowledge base construction (Craven et al., 1998; Craven and Kumlien, 1999).

One problem in NER is the diversity of entity types, which vary in scope for different domains and downstream tasks. Traditional NER for the news domain focuses on three coarse-grained entity types: *person*, *location*, and *organization* (Tjong Kim Sang and De Meulder, 2003). However, as NLP technologies have been applied to a broader set of domains, many other entity types have been targeted. For instance, Ritter et al. (2011) add seven new entity types (e.g., *product*, *tv-show*) on top of the previous three when annotating tweets. Other efforts also define different but partially overlapping sets of entity types (Walker et al., 2006; Ji et al., 2010; Consortium, 2013; Aguilar et al., 2014). These non-unified annotation schemas result in *partially annotated datasets*: each dataset is only annotated with a subset of possible entity types.

One approach to this problem is to train individual NE taggers for each partially annotated dataset and combine their results using some heuristics. Figure 1 shows an example that demonstrates the possible shortcomings of this approach, using the biomedical domain as a case study.[2] Here, we train four separate models on four partially annotated datasets: AnatEM (Pyysalo and Ananiadou, 2013) annotated for the *anatomy* type, BC2GM (Smith et al., 2008) for the *gene* type, JNLPBA (Kim et al., 2004) for *cell* types, and Linnaeus (Gerner et al., 2010) for the *species* type. We can see that the models' predictions contradict each other when applied to the same test sentence—making it a challenge to accurately combine them.

In this paper, we propose a deep structured model to leverage multiple partially annotated datasets, allowing us to jointly identify the union

* Work done while the author was at USC ISI.

[1]The code and the datasets will be made available at https://github.com/xhuang28/NewBioNer

[2]https://corposaurus.github.io/corpora/ summarizes dozens of partially annotated biomedical datasets.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 515–527
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

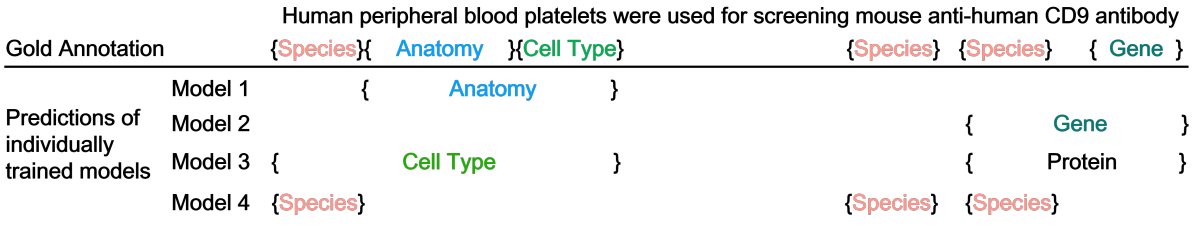


Figure 1: An example sentence from the CellFinder corpus (Neves et al., 2012) showing the challenges in combining the output of individual NE taggers. The *Gold* row is the human annotations in CellFinder. The rows below are predictions made by models trained on datasets that each contain only a subset of the CellFinder types. Note that the individual taggers' predictions can conflict with each other, making it challenging to combine them. (Note: we renamed CellFinder's *Cell Component* to *Cell Type* to fit it in the space above.)

of all entity types presented in the training data. The model leverages supervision signals across diverse datasets to learn robust input representations, thus improving the performance for each entity type. Moreover, it makes *joint* predictions to avoid potential conflicts among models built on different entity types, allowing further improvement for cross-type NER.

Experiments on both real-world and synthetic datasets show that our model can efficiently adapt to new corpora that have more types than any individual dataset used for training and that it achieves significantly better results compared to strong multi-task learning baselines.

2 Problem Statement

We formally define the problem by first defining our terminology.

Global Tag Space. Let C_i denote a corpus, and T_{C_i} denote the set of entity types that are tagged in corpus C_i. When there are a set of corpora $\mathbb{C} = \{C_1, C_2, ..., C_n\}$, each has its own tag space concerning different entity types, the global tag space is defined as the union of the local tag space. Formally, $T_{\mathbb{C}} = T_{C_1} \cup T_{C_2} \cup ... \cup T_{C_n}$.

Partially Annotated Corpus. If $T_{C_i} \subsetneqq T_{\mathbb{C}}$, then C_i is a partially annotated corpus.

Global Evaluation. When a model is trained on a set of partially annotated corpora $\mathbb{C}$ and predicts tags for the whole global tag space $T_{\mathbb{C}}$, we say it is making *global predictions*. Accordingly, the evaluation of the models' performance on $T_{\mathbb{C}}$ is called *global evaluation*.

Our goal is to train a single unified NE tagger from several *partially annotated* corpora for efficient adaptation to new corpora that have more types than any individual dataset used during training. Formally, we have a set of corpora $\mathbb{C} = \{C_1, C_2, ..., C_n\}$, and we propose to train a joint model on $\mathbb{C}$ such that it makes predictions for the *global tag space* $T_{\mathbb{C}}$. One benefit of this joint model is that it can be easily adapted to a new tag space T_{C_u} where $T_{C_u} \subseteq T_{\mathbb{C}}$, and $T_{C_u} \nsubseteq T_{C_i}, \forall C_i \in \mathbb{C}$.

3 Background and Related Work

In this section, we first introduce neural architectures for NER which our work builds upon and then summarize previous work on imperfect annotation problems.

3.1 Neural Architectures for NER

With recent advances using deep neural networks, bi-directional long short-term memory networks with conditional random fields (BiLSTM-CRF) have become standard for NER (Lample et al., 2016). A typical architecture consists of a BiLSTM layer to learn feature representations from the input and a CRF layer to model the interdependencies between adjacent labels and perform joint inference. Ma and Hovy (2016) introduce additional character-level convolutional neural networks (CNNs) to capture subword unit information. In this paper, we use a BiLSTM-CRF with character-level modeling as our base model. We now briefly review the BiLSTM-CRF model.

BiLSTMs. Long Short Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) are a variation of RNNs that are designed to avoid the vanishing/exploding gradient problem (Bengio et al., 1994). Specifically, BiLSTMs take as input a sequence of words $\boldsymbol{x} = \{x_k | k \in \mathcal{N}\}$ and output a sequence of hidden vectors: $\boldsymbol{H} = \{h_k | k \in \mathcal{N}\}$ BiLSTMs combine a left-to-right (forward) and a right-to-left (backward) LSTM to capture both left and right context. Formally, they produce a hidden vector $\tilde{\boldsymbol{h}}_i = [\overrightarrow{\boldsymbol{h}_i}; \overleftarrow{\boldsymbol{h}_i}]$ for each input, where $\overrightarrow{\boldsymbol{h}_i}$ and

$\overleftarrow{\boldsymbol{h}_i}$ are produced by the forward and the backward LSTMs respectively; $[;]$ denotes vector concatenation.

Character-level Modeling. Following Wang et al. (2018), we use a BiLSTM for character-level modeling. We concatenate the hidden vector of the space after a word from the forward LSTM and the hidden vector of the space before a word from the backward LSTM to form a character-level representation of the word: $\boldsymbol{h}_i^c = [\overrightarrow{\boldsymbol{h}_i^c}; \overleftarrow{\boldsymbol{h}_i^c}]$. The word-level BiLSTM then takes the concatenation of $\boldsymbol{h}_i^c$ and the word embedding as input $\boldsymbol{x}_i = [\boldsymbol{e}_i; \boldsymbol{h}_i^c]$ to learn contextualized representations.

Neural-CRFs. Conditional Random Fields (CRFs) (Lafferty et al., 2001) are sequence tagging models that capture the inter-dependencies between the output tags; they have been widely used for NER (McCallum and Li, 2003; Lu et al., 2015; Peng and Dredze, 2015, 2016, 2017). Given a set of training data $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}^N$, a CRF minimizes negative log-likelihood:

$$\min_{\Theta} -\sum_i \log P(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \Theta), \tag{1}$$

$$P(\boldsymbol{y}_i \mid \boldsymbol{x}_i; \Theta) = \frac{Gold\,Energy}{Partition} = \frac{St(\boldsymbol{y}_i)}{\sum_{\boldsymbol{y}'} St(\boldsymbol{y}')} \tag{2}$$

where $\boldsymbol{y}'$ is any possible tag sequence with the same length as $\boldsymbol{y}_i$, $St(\boldsymbol{y}')$ is the potential of the tag sequence $\boldsymbol{y}'$, and $St(\boldsymbol{y}_i)$ is the potential of the gold tag sequence. The numerator $St(\boldsymbol{y}_i)$ is called the *gold energy function*, and the denominator $\sum_{\boldsymbol{y}'} St(\boldsymbol{y}')$ is the *partition function*. The likelihood function using globally annotated data is illustrated in Figure 2a. The potential of a tag sequence can be computed as:

$$St(\boldsymbol{y}) = \prod_{t=1}^{|\boldsymbol{y}|} Score(\boldsymbol{y}[t], \boldsymbol{y}[t-1]) \tag{3}$$

where $\boldsymbol{y}[t]$ is the tth element in $\boldsymbol{y}$ ($\boldsymbol{y}[-1]$ is the start of the sequence), and

$$Score(\boldsymbol{y}[t], \boldsymbol{y}[t-1]) = \exp\left(tr(\boldsymbol{y}[t], \boldsymbol{y}[t-1])\right) * \exp\left(em(\boldsymbol{y}[t])\right) \tag{4}$$

where $tr(\boldsymbol{y}[t], \boldsymbol{y}[t-1])$ is the transition score from $\boldsymbol{y}[t-1]$ to $\boldsymbol{y}[t]$, and $em(\boldsymbol{y}[t])$ is the emission score of $\boldsymbol{y}[t]$ computed based on the output $\tilde{\boldsymbol{h}}_t$ of the BiLSTM.

3.2 Learning from Imperfect Annotations

Learning from multiple partially annotated datasets could be more generally thought of as learning from imperfect annotations. In that broad sense, there are several notable areas of prior work. One of the most prominent concerns learning from *incomplete annotations* (noisy labels), where some occurrences of entities are neglected in the annotation process and falsely labeled as non-entities (negative). A related problem is learning from *unlabeled data* with distant supervision.

A major challenge of all these settings, including ours, is that a positive instance might be labeled as negative. A well-explored solution to this problem is proposed by Tsuboi et al. (2008), which instead of maximizing the likelihood of the gold tag sequence, we maximize the total likelihood for all possible tag sequences consistent with the gold labels. Tsuboi et al. (2008); Yang and Vozila (2014) applied this idea to the *incomplete annotation* setting; Shang et al. (2018); Liu et al. (2014) applied it to the *unlabeled data* with distant supervision setting; and Greenberg et al. (2018) applied it to the *partial annotation* setting. While this is a general solution, its primary drawback is that it assumes a uniform prior on all labels consistent with the gold labels. This may have the result of overly encouraging the prediction of entities, resulting in low precision.

To tackle the problem of incomplete annotations, Carlson et al. (2009); Yang et al. (2018) explored bootstrap-based semi-supervised learning on unlabeled data, iteratively identifying new entities with the taggers and then re-training the taggers. Bellare and McCallum (2007); Li and Liu (2005); Fernandes and Brefeld (2011) explored an EM algorithm with semi-supervision.

For the *partial annotation* problem, most previous work has focused on building individual taggers for each dataset and using single-task learning (Liu et al., 2018) or multi-task learning (Crichton et al., 2017; Wang et al., 2018). In single-task learning, each model is trained separately on each dataset C_i, and makes local predictions on T_{C_i}. Based on the neural-CRF architecture, multi-task learning uses a different CRF layer for each dataset C_i (each task) to make local predictions on T_{C_i}, and shares the lower-level representation learning component across all tasks. Both single-task learning and multi-task learning make local

predictions and have to apply heuristics to combine the model predictions, resulting in the collision problem demonstrated in Figure 1.

To the best of our knowledge, Greenberg et al. (2018) is the only prior work trying to build a unified model from multiple partially annotated corpora. We will show that their model, which is reminiscent of Tsuboi et al. (2008), is a special case of ours and that our other variations achieve better performance. In addition, they only evaluated the model on the training corpora while we conduct evaluations to test the model's ability to adapt to new corpora with different tag spaces.

4 Model

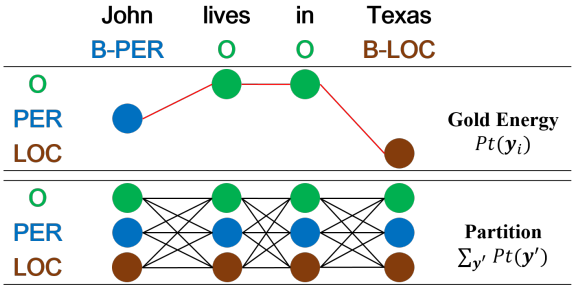
As mentioned above, we use a BiLSTM-CRF with character-level modeling as our base model. Our goal is to build a unified model to make global predictions. That is, our model will be jointly trained on multiple partially annotated datasets $\mathbb{C}$ and make predictions on the global tag space $T_{\mathbb{C}}$. Such a unified model will enjoy the benefit of learning robust representations from multiple datasets just like multi-task learning while maintaining a joint probability distribution of the global tag space to avoid possible conflicts from individual models.

4.1 Naive Approach

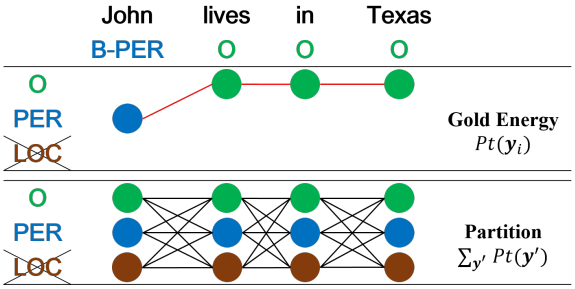
A simple solution to the problem is to merge all the datasets into one giant corpus. A single model can then be trained on this corpus to make global predictions. However, such a corpus will be missing many correct annotations, since each portion will be annotated with only a subset of the target entity types. Figure 2b shows an example: here, a location (*Texas*) exists but is labeled as a non-entity, because the original dataset from which this sentence is drawn does not annotate locations at all. As a result, this approach suffers from *false penalties* when applying the original likelihood function (Eq. 2-4) to train the model, meaning that it penalizes predictions that correctly identify entities with types that are not annotated for a particular sentence.

4.2 Improving the Gold Energy Function

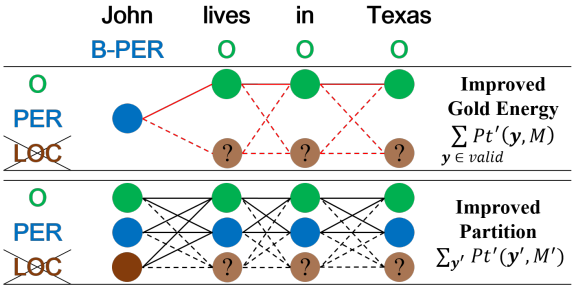
One way to improve performance is to explicitly acknowledge the incompleteness of the existing "gold" annotations and to give the model credit for predicting any tag sequence that is consistent with



(a) **Original likelihood function** with *global annotation*



(b) **Original likelihood function** with *partial annotation*



(c) **Improved likelihood function** with *partial annotation*

Figure 2: Illustration of original (2a, 2b) and improved (2c) likelihood functions. Each figure has two parts upper and lower that illustrate the gold energy (numerator) and the partition (denominator) respectively. Solid lines represent tag sequences that are fully considered in the functions. Dashed lines represent tag sequences that are discounted. The sentences in 2b and 2c are not annotated with **LOC**.

the partial annotations. This can be done by modifying the CRF's gold energy function, illustrated in the upper part of Figure 2c. Specifically, in this example, *John* is labeled as *PER*, so *PER* is the only possible correct tag at that position. However, *lives*, *in*, and *Texas* are labeled as *O* (non-entity), which here means only that they may not be *PER*—but any of them could be *LOC*, since locations are not annotated for this sentence. Therefore, any sequence that assigns either *O* or *LOC* for any of these three positions is consistent with the gold labels. To account for this, we modify the gold energy function to credit all tag sequences

that are consistent with the gold annotations, encouraging the model to predict other consistent labels when the gold label is *O*. Tsuboi et al. (2008) propose a specific solution that applies this idea on incomplete annotations: instead of maximizing the likelihood of the gold tag sequence when optimizing the CRF model, they maximize the total likelihood of all possible tag sequences consistent with the gold labels. This approach is later used by Greenberg et al. (2018) to handle the problem of partial annotation. We will address a potential problem with their method and propose a generalized version in Section 4.4.

4.3 Improving the Partition Function

Modifying the gold energy function will give credit to a system for producing alternative entity labels for words tagged as *O* in the partially annotated training. A different solution is to simply *not penalize* predictions of such alternative labels. This can be done by modifying the partition function and keeping the gold energy function unchanged. The lower part of Figure 2c gives an illustration. As stated above, *LOC* is a consistent alternative entity label for *lives*, *in*, and *Texas*. We therefore exclude from our calculations any paths that include *LOC* at any of those positions. More generally, we exclude all such consistent but alternative tag sequences from the computation of the CRF's partition function. Section 4.4 gives formal definitions with equations. The improved partition function sets the model free to predict alternative labels without penalty (as long as they are consistent with the known gold annotations), but it does not give them any positive credit for doing so (as in the previous approach). We hypothesize that the improved partition function would work better than the improved gold energy function in our setting because it addresses the *false penalties* problem more precisely. We will verify this hypothesis in our experiments.

4.4 Discounting Alternative Sequences

There is a potential problem with naively applying the improved gold energy function: when the gold label is *O*, the model is encouraged to predict other consistent labels as strongly as it is encouraged to predict *O*. However, many *O* labels are confident annotations of *O*. As a result, naively training with the improved gold energy function tends to over-predict entities and not predict *O*s. To mitigate this problem, we discount the energy of tag sequences that go through alternative labels. This can be achieved by introducing a hyper-parameter $M(\text{mask}) \in [0, 1]$ as a discounting factor for the gold energy function. Formally, we modify Eq 3 to:

$$St'(\boldsymbol{y}, M) = \prod_{t=1}^{|\boldsymbol{y}|}(Score(\boldsymbol{y}[t], \boldsymbol{y}[t-1]) * mask(\boldsymbol{y}[t], M)),$$

where

$$mask(\boldsymbol{y}[t], M) = \begin{cases} M, & \text{if } \boldsymbol{y}[t] \in alternative \\ 1, & \text{Otherwise} \end{cases}.$$

where $alternative$ is the set of alternative labels. We thus have the improved gold energy function:

$$Improved\,Gold\,Energy = \sum_{\boldsymbol{y} \in valid} St'(\boldsymbol{y}, M), \quad (5)$$

where $valid$ is the set of all tag sequences that are consistent with the gold sequence, including the gold sequence itself.

Similarly, for the improved partition function, we can use the same strategy to discount the energy of alternative sequences rather than completely removing them. We thus introduce another $M' \in [0, 1]$ and the improved partition function becomes:

$$Improved\,Partition = \sum_{\boldsymbol{y}'} St'(\boldsymbol{y}', M'), \quad (6)$$

4.5 Combining Improved Functions

For generality, we combine the improved gold energy and the improved partition function to make a new likelihood function as our final model:

$$Improved\,LH = \frac{\sum_{\boldsymbol{y} \in valid} St'(\boldsymbol{y}, M)}{\sum_{\boldsymbol{y}'} St'(\boldsymbol{y}', M')} \quad (7)$$

To ensure Equation 7 is a valid likelihood function (the probabilities of all sequences sum to 1), we need a constraint that $M = M'$. Note that Equation 7 subsumes all models discussed in this section. Specifically, when $M = 0, M' = 1$, the model is the *Naive Model* discussed in Section 4.1; when $M = 1, M' = 1$, the model is the same as Greenberg et al. (2018) discussed in Section 4.2; when $M = 0, M' = 0$, the model

is the same as proposed in Section 4.3. We have a general perspective of all the models by simply treating M and M' as hyper-parameters.

Note that for the *Naive Model*, since $M! = M$, the Equation 7 is not always a valid likelihood function[3]. This may partially explain why the *Naive Model* performs so poorly under this setting. We posit that the model will work the best when $M = M'$.

5 Experimental Setup

5.1 Datasets

Our goal is to train a unified NER model on multiple partially annotated datasets. This model will make global predictions and can efficiently adapt to new corpora that contain tags from more than one training corpus. To fully test this capability, we would need a single test set annotated with all types of interest. However, the motivation behind this effort is that such a dataset typically does not exist. We therefore take two approaches to approximate such an evaluation.

In the first evaluation setting, we take advantage of the fact that although there may not be a *single* dataset annotated with all types of named entities of interest, there exist several datasets that cover types from more than one of the training corpora. Specifically, we are able to select test corpora that each cover types of interest from multiple training corpora. Table 1 shows the biomedical corpora we use and their entity types. For example, we use **BC5CDR** for global evaluation, because its entity types (*Chemical* and *Disease*) cover multiple training corpora (**BC4CHEM** for *Chemical* and **NCBI** for *Disease*).

In the second evaluation setting, we create synthetic datasets from the CoNLL 2003 NER dataset to simulate training and global evaluations. Specifically, the CoNLL 2003 dataset is annotated with four entity types: location, person, organization, and miscellaneous entities. We randomly split the training set into four portions, each containing only one entity type (all other types are removed). In this setting, the four portions of the training set are used for training and the original dataset with all entities annotated is used as a global corpus.

More details about all the datasets can be found in Appendix A.1.

[3]This may be confusing because when $M = 0, M' = 1$ it looks exactly the same as the original CRF likelihood function. But in the partial annotation setting, this means that the scores of alternative sequences will be zero in the numerator but non-zero in the denominator, which makes the total likelihood less than 1. It suggests that the original CRF likelihood function is not suitable for the partial annotation setting.

For Training		For Global Evaluations	
Corpus	**Entities**	**Corpus**	**Entities**
BC2GM	GP	BC5CDR	Chemical, Disease
BC4CHEM	Chemical		
NCBI	Disease	BioNLP13CG	GP, Disease, Chemical, others
JNLPBA	GP, DNA, Cell-type, Cell-line, RNA	BioNLP11ID	GP, Chemical, others
Linnaeus	Species		

Table 1: Details of the biomedical corpora. "others" denotes NE types that do not appear in the training corpora, and thus are not evaluated.

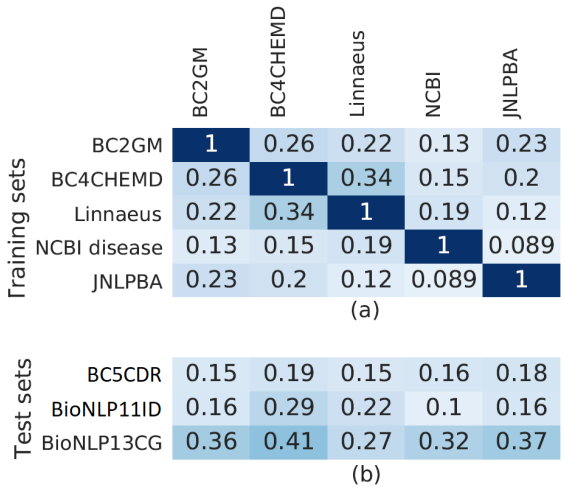


Figure 3: (a) The mention-level overlap among training sets. (b) The mention-level overlap between training datasets and evaluation datasets.

5.1.1 Biomedical Dataset Analysis

The motivation for this work rests on the assumption that even when a dataset is annotated for a certain set of entity types, it likely contains other types of entities that are unlabeled. To verify this assumption, we expand the annotations of each dataset using heuristics and compute the pairwise mention-level overlap between the datasets. Specifically, suppose we are comparing two datasets, A and B. We first construct A' and B', where A' contains all mentions in A but is augmented with new mentions found by taking all strings annotated in B and marking them as named entities in A (regardless of context; there may obviously be some errors). We do the same (in the

opposite direction) to construct B'. We then compute the pairwise overlap coefficient between A' and B' according to the following criterion:

$$\text{overlap}(A', B') = \frac{|A' \cap B'|}{\min(|A'|, |B'|)}.$$

Figure 3 shows the heat maps. For the training group, *BC2GM*, *BC4CHEMD*, and *Linnaeus* are considerably overlapped, although they are annotated with different entity types (GP, Chemical, and Species). This confirms our assumption that although the datasets are annotated for a subset of entity types, they contain other types that are unlabeled.[4]

5.2 Hyper-parameters.

We borrow most of the best hyper-parameters reported by Wang et al. (2018). The hidden sizes of the BiLSTMs are tuned, and the best value we found is 100 for the character-level BiLSTM, and 300 for the word-level BiLSTM. We also tuned both discounting factors M and M' in the range of [0,0.2,0.4,0.6,0.8,1.0]. It turns out that $M = 0, M' = 0$ (using improved partition function) and $M = 1, M' = 1$ (using improved gold energy function) make two local optimums. Therefore we report the performance of three special cases of our proposed framework, with $M, M' = [0, 0]$, $[1, 1]$, and $[0, 1]$ (the naive model), respectively.

5.3 Compared Models.

We compare different variations of our unified model and other models in different settings. We first train models on all training corpora, and then perform evaluations under two scenarios: (1) **no-supervision**: directly evaluating the trained models on each global corpora; (2) **limited-supervision**: fine-tuning the models on a small subset of the training portion of each global corpus before the evaluations.

Under both scenarios, we report performance of four different models:

- **MTM/MTM-vote**: Train a multi-task model (MTM) on training corpora, using a separate CRF for each corpus. (This is the current state-of-the-art structure (Wang et al., 2018) when evaluated on the training corpora.)
  - Under the no-supervision setting, we heuristically combine all existing CRF's predictions to make global predictions. Specifically, we apply two heuristics to resolve conflicts while preserving entity chunk-level consistency. First, where predictions from more than one model overlap, we expand each prediction's boundary to the outermost position. Second, we always favor the predictions of named entities over the predictions of non-entity.[5]
  - Under the limited-supervision setting, for each global corpus, we add a new CRF and train it along with the LSTMs.
- **Unified-01**: Use the naive training approach described in 4.1; this corresponds to our unified model with settings $M = 0, M' = 1$.
- **Unified-11**: Use the improved gold energy function described in 4.2; this corresponds to our unified model with settings $M = 1, M' = 1$ and is equivalent to the model proposed by Greenberg et al. (2018).
- **Unified-00**: Use the improved partition function proposed in 4.3; this corresponds to our unified model with settings $M = 0, M' = 0$.

Among the compared models, *Unified-01* (the naive model) and *MTM/MTM-Vote* are either simple or commonly used methods and thus are treated as baselines. *Unified-00* is a novel approach. Although Greenberg et al. (2018) used the approach of *Unified-11*, they only evaluated the model on training corpora/tasks while we apply it for task adaptation. Moreover, it is a special case of our proposed framework, thus we argue that people can simply tune M and M' to get good performance for adaptations to new tasks.

6 Results

As mentioned above, we compare the results of four different approaches in no-supervision and limited-supervision settings, both with real-world biomedical data and synthetic news data.

As a sanity check, we also evaluate the models on the test sets of the training corpora. The results can be found in Appendix A.2. It is shown that

[4] We further verified this conclusion by computing the heat maps on the original datasets. The overlaps between *BC2GM* and *BC4CHEMD*, and *BC2GM* and *Linnaeus* are nearly 0.

[5] A lower recall and f1 score was observed in the initial experiment without this heuristic.

Corpus	Trained on Other Biomedical Datasets									Traind on CoNLL		
	BC5CDR			BioNLP13CG			BioNLP11ID			CoNLL 2003		
	F	P	R	F	P	R	F	P	R	F	P	R
MTM-Vote	63.6	64.4	62.8	61.0	56.7	65.9	50.4	44.8	57.5	83.9	88.4	79.8
Unified-01	42.7	93.7	27.6	37.5	72.5	25.3	23.6	50.8	15.4	01.6	97.8	00.8
Unified-11	70.2	73.8	67.0	67.7	64.0	71.9	53.2	47.1	61.1	80.1	84.6	76.1
Unified-00	**73.8**	84.1	65.7	**69.7**	68.1	71.5	52.7	49.4	56.5	**84.8**	90.0	80.2

Table 2: Results for task adaptation in the no-supervision setting. The best f1 score in each column that is significantly better than the second best is bold-faced, while those are better but not significantly are underlined. All the significance tests are conducted using mention-level McNemar's Chi-square test, with p-value = 0.01.

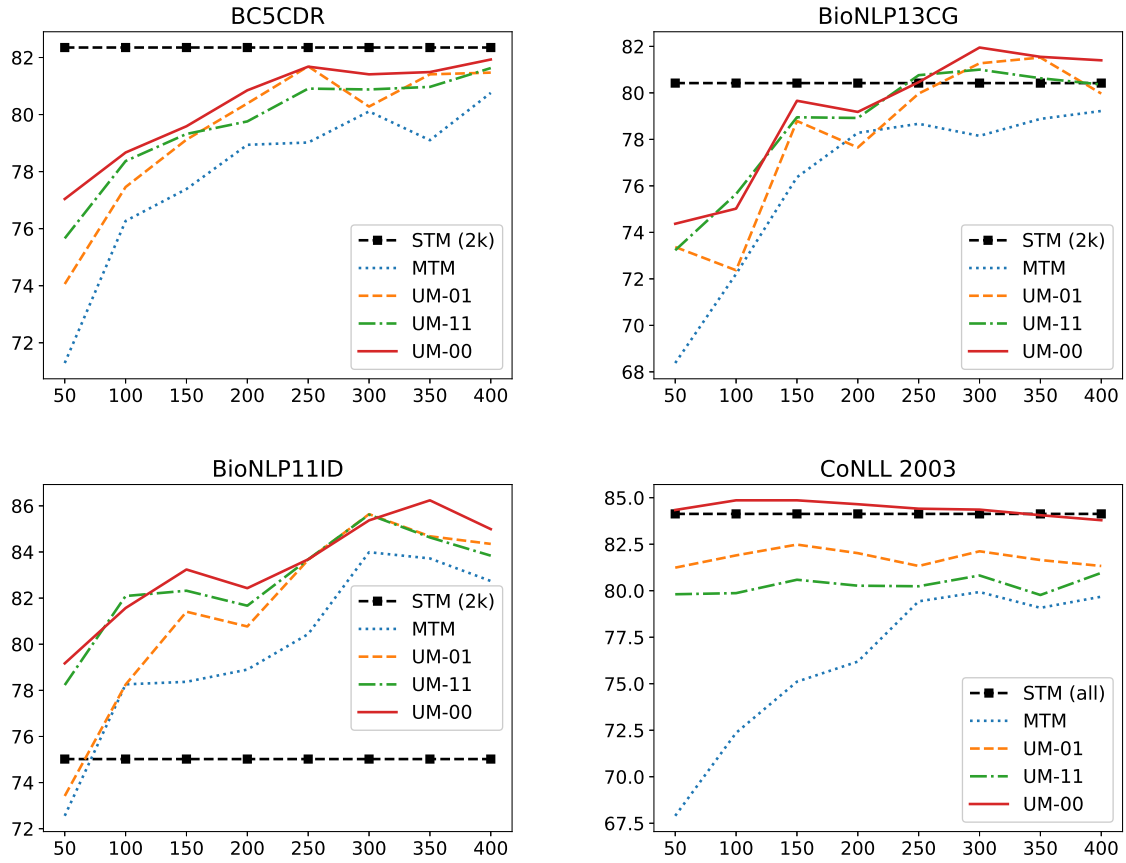


Figure 4: Plot of f1 scores for task adaptation in the limited-supervision setting. X-axis represents the number of sentences used for fine-tuning. *STM(2k)* is a STM trained on 2k sentences sampled from the global corpus, and *STM(all)* is trained on the entire training set of the corpus.

our MTM performs comparably with state-of-the-art systems evaluated on the training corpora, and thus is a strong baseline.

6.1 No-Supervision Setting

Table 2 demonstrates the results for task adaptation in the no-supervision setting. We report precision and recall in addition to f1 scores to better show the differences between the models.

Comparing on f1 scores, *Unified-00* (our new model) significantly outperforms all other models on three out of four datasets, demonstrating its effectiveness. *Unified-11* also achieves good results, with higher recall but lower precision than *Unified-00*. This aligns well with our hypothesis that it encourages predictions of entities. Conversely, *Unified-01* (the naive approach) achieves the highest precision but lowest recall, which is reasonable considering the problem of false penalties that discourages the model from predicting en-

tities. We also found that the model achieves better performance when $M = M'$, which supports our hypothesis in 4.5 that the model works better with a valid likelihood function.

6.2 Limited-Supervision Setting

To further demonstrate the models' ability to adapt to new datasets with a small amount of supervision, we sample a small subset of the training portion of each global evaluation corpus to fine-tune the trained models. We show the performance of the models fine-tuned with different amounts of sampled data. For each global corpus, we show a single-task model (STM) trained on it with a reasonable amount of data (two thousand sentences for the biomedical corpora). In the CoNLL 2003 setting, we train the STM on the entire training data for a fair comparison, because all other models are first trained on the four training portions, which essentially look through the entire training set (just partially annotated). The results of the STMs are used as benchmarks. Experimental results are presented in Figure 4.

Firstly, with much less training data, all the models achieve comparable or noticeably better performance than the STMs trained from scratch, demonstrating that training on the partially annotated corpora does help to boost performance on global evaluation corpora. Additionally, MTMs are worse than all the unified models, because they only share the LSTM layers, but lose all the knowledge in the CRFs when adapted to new corpora. The unified models have the advantage that they can reuse the robust CRFs learned from a large amount of data. This is more obvious in the CoNLL 2003 evaluation setting, where the unified models that reuse the pre-trained CRFs achieve good performance trained with only 50 sentences, but the MTM, which does not reuse the CRFs, needs a larger amount of training data to catch up.

In general, *Unified-00*, our novel approach proposed here, still performs the best on every dataset. We note that although *Unified-01* has an extremely low recall on the CoNLL 2003 dataset in the no-supervision setting, it works surprisingly well in the limited-supervision setting. On the other hand, *Unified-00* and *Unified-11* generally perform better than *Unified-01* on real-world biomedical datasets, especially when fine-tuned on less data. Again, since all the unified models are special cases of our proposed framework, we argue that, for adapting to new datasets, people can simply tune the discounting factors M and M' to get good results.

7 Conclusion and Future Work

In this paper, we propose a unified model that learns from multiple partially annotated datasets to make *joint* predictions on the union of entity types appearing in any training dataset. The model integrates learning signals from different datasets and avoids potential conflicts that would result from combining independent predictions from multiple models. Experiments show that the proposed unified model can efficiently adapt to new corpora that have more entity types than any of the training corpora, and performs better than the baseline approaches.

In future work, we plan to explore other algorithms (e.g. imitation learning) that allow the model the explore the unknown space during training, using delayed rewards to decide whether the model should trust its exploration. Analysis of the global evaluation results suggests that the unified model is under-predicting, meaning there is still room for improvement specifically on recall. We plan to explore further changes to the current objectives to encourage more entity predictions.

Finally, the approach proposed in this paper also does not handle entity types of varying granularities or tagsets with mismatched guidelines (e.g. one dataset annotates only for-profit companies as *ORG* and one annotates all formalized groups). Effectively modeling these complications is an interesting area for future work.

Acknowledgements

We thank the anonymous reviewers for their constructive comments, as well as the members of the USC PLUS lab for their early feedbacks. We thank Tianyu Meng and Yuxin Zhou for their help with initial data processing and experimental setup. This work is supported in part by DARPA (HR0011-15-C-0115) and an NIH R01 (LM012592). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the sponsors.

References

Jacqueline Aguilar, Charley Beller, Paul McNamee, Ben V. Durme, Stephanie Strassel, Zhiyi Song, and Joe Ellis. 2014. A comparison of the events and relations across ACE, ERE, TAC-KBP, and FrameNet annotation standards. In *ACL Workshop: EVENTS*.

Kedar Bellare and Andrew McCallum. 2007. Learning extractors from unlabeled text using relevant databases. In *Sixth international workshop on information integration on the web*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*.

Andrew Carlson, Scott Gaffney, and Flavian Vasile. 2009. Learning a named entity tagger from gazetteers with the partial perceptron. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*.

Linguistic Data Consortium. 2013. DEFT ERE annotation guidelines: Relations v1.1. *Linguistic Data Consortium, Philadelphia*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*.

Mark Craven, Andrew McCallum, Dan PiPasquo, Tom Mitchell, and Dayne Freitag. 1998. Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie-mellon univ pittsburgh pa school of computer Science.

Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. 2017. A neural network multi-task learning approach to biomedical named entity recognition. *BMC bioinformatics*.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*.

Eraldo R Fernandes and Ulf Brefeld. 2011. Learning from partially annotated sequences. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 407–422. Springer.

Martin Gerner, Goran Nenadic, and Casey M Bergman. 2010. Linnaeus: a species name identification system for biomedical literature. *BMC bioinformatics*.

Nathan Greenberg, Trapit Bansal, Patrick Verga, and Andrew McCallum. 2018. Marginal likelihood training of bilstm-crf for biomedical named entity recognition from disjoint label sets. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2824–2829.

Takaaki Hasegawa, Satoshi Seki, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *ACL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *TAC*.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at jnlpba. In *JNLPBA*. ACL.

Jin-Dong Kim, Yue Wang, and Yamamoto Yasunori. 2013. The genia event extraction shared task, 2013 edition-overview. In *BioNLP Shared Task 2013 Workshop*.

Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. 2015. Chemdner: The drugs and chemical names extraction challenge. *Journal of Cheminfo.*

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL*.

Xiao-Li Li and Bing Liu. 2005. Learning from positive and unlabeled examples with different data distributions. In *ECML*.

L. Liu, J. Shang, F. Xu, X. Ren, H. Gui, J. Peng, and J. Han. 2018. Empower Sequence Labeling with Task-Aware Neural Language Model. In *AAAI*.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for crf-based chinese word segmentation using free annotations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 864–874. Association for Computational Linguistics.

Yanan Lu, Donghong Ji, Xiaoyuan Yao, Xiaomei Wei, and Xiaohui Liang. 2015. Chemdner system with mixed conditional random fields and multi-scale word clustering. *Journal of cheminformatics*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *NAACL*.

Raymond J Mooney and Razvan C Bunescu. 2005. Subsequence kernels for relation extraction. In *NIPS*.

Mariana Neves, Alexander Damas, Andreas Kurtz, and Ulf Leser. 2012. Annotating and evaluating text for stem cell research. In *BioTxtM workshop at LREC on Building and Evaluation Resources.*

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisboa, Portugal.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media via learning segmentation representations. In *ACL*.

Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Proceddings of the ACL Workshop on Representation Learning for NLP*.

Sampo Pyysalo and Sophia Ananiadou. 2013. Anatomical entity mention recognition at literature scale. *Bioinformatics*.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *EMNLP*.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Jingbo Shang, Liyuan Liu, Xiang Ren, Xiaotao Gu, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. *arXiv preprint arXiv:1809.03599*.

Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, et al. 2008. Overview of biocreative ii gene mention recognition. *Genome biology*.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL at HLT-NAACL*.

Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 897–904. Coling 2008 Organizing Committee.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. *LDC*.

Xuan Wang, Yu Zhang, Xiang Ren, Yuhao Zhang, Marinka Zitnik, Jingbo Shang, Curtis Langlotz, and Jiawei Han. 2018. Cross-type biomedical named entity recognition with deep multi-task learning. *CoRR*.

Chih-Hsuan Wei, Yifan Peng, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Jiao Li, Thomas C Wiegers, and Zhiyong Lu. 2015. Overview of the biocreative v chemical disease relation (cdr) task. In *BC V Workshop*.

Fan Yang and Paul Vozila. 2014. Semi-supervised chinese word segmentation using partial-label learning with conditional random fields. In *EMNLP*.

Yaosheng Yang, Wenliang Chen, Zhenghua Li, Zhengqiu He, and Min Zhang. 2018. Distantly supervised ner with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2159–2169.

Corpus	Named Entities	Sents	Tokens	Mentions
BC2GM	Gene/Protein	20,131	569,912	24,585
BC4CHEM	Chemical	87,685	2,544,305	84,312
NCBI	Disease	7,287	184,167	6,883
JNLPBA	Gene/Protein, DNA, Cell-type, Cell-line, RNA	24,806	595,994	59,965
Linnaeus	Species	23,155	539,428	4,265

Table 3: Statistics for the Training Corpora

Corpus	Named Entities	Sents	Tokens	Mentions
BC5CDR	Chemical, Disease	13,938	360,373	28,789
BioNLP13CG	Gene/Protein, Disease, Chemical, Others	1,906	52,771	6881
BioNLP11ID	Gene/Protein, Chemical, Others	5178	166416	11084

Table 4: Statistics for global evaluation corpora. "Others" denote the NEs which do not appeared in training data, thus are not evaluated.

A Appendix

A.1 Datasets

Below we introduce the datasets in the biomedicine domain and the news domain.

A.1.1 Biomedicine domain: Local training group

The training group consists of five datasets: *BC2GM*, *BC4CHEM*, *NCBI-disease*, *JNLPBA*, and *Linnaeus*. The first two datasets are from different BioCreative shared tasks (Smith et al., 2008; Krallinger et al., 2015; Wei et al., 2015). *NCBI-disease* is created by Doğan et al. (2014) for disease name recognition and normalization. *JNLPBA* comes from the 2004 shared task from joint workshop on natural language processing in biomedicine and its applications (Kim et al., 2004), and *Linnaeus* is a species corpus composed by Gerner et al. (2010). More information about the datasets can be found in Table 3.

Below are detailed descriptions of the datasets:

BC2GM is a gene/protein corpus. The annotation is Gene. It's provided by the BioCreative II Shared Task for gene mention recognition.

BC4CHEM is a chemical corpus. The annotation is Chemical. It's provided by the BioCreative IV Shared Task for chemical mention recognition.

	Articles	Sentences	Tokens
Training set	946	14,987	203.621
Development set	216	3,466	51,362
Test set	231	3,684	46,435

Table 5: Statistics for the CoNLL 2003 NER dataset

NCBI-disease is a disease corpus. The annotation is Disease. It was introduced for disease name recognition and normalization.

JNLPBA consists of DNA, RNA, Gene/Protein, Cell line, Cell Type. The annotation is same as the NE names, except the Gene/Protein is annotated with Protein. It was provided by 2004 JNLPBA Shared Task for biomedical entity recognition.

Linnaeus is a species corpus. The annotation is Species. The original project was created for entity mention recognition.

A.1.2 Biomedicine domain: Global evaluation group

We reemphasize here that the purpose of the global evaluation is to test the model's ability to making global predictions and efficiently adapt to global corpora. While no corpus is globally annotated, we identify several existing corpora to *approximate* the global evaluation. Each test corpus is annotated with a *superset* of several training corpora to test the model's generalizability outside of the local tag spaces.

The global evaluation group contains three datasets: *BC5CDR*, *BioNLP13CG*, and *BioNLP11ID*. Each is annotated with multiple entity types. *BC5CDR* comes from the BioCreative shared tasks (Smith et al., 2008; Krallinger et al., 2015; Wei et al., 2015). *BioNLP13CG* and *BioNLP11ID* come from the BioNLP shared task (Kim et al., 2013). More information about the global evaluation datasets can be found in Table 4.

Below are detailed descriptions of the datasets:

BC5CDR is a chemical and disease corpus. The annotation is Chemical and Disease. It's provided by BioCreative V Shared Task for chemical and disease mention recognition.

BioNLP13CG consists of Gene/Protein and Related Product, Cancel, Chemical, Anatomy and Organism and others. **BioNLP11ID** consists of Gene/Protein, Chemical, and Organism. The annotation is same as the NE types but has a finer ontology scope.

Corpus	BC2GM	BC4CHM	NCBI	JNLPBA	Linnaeus
STM	79.9	88.6	84.1	72.7	87.3
MTM Crichton et al. (2017)	73.2	83.0	80.4	70.1	84.0
MTM Wang et al. (2018)	80.7	89.4	86.1	73.5	-
MTM (ours)	80.3	89.2	85.8	73.5	**88.5**
Unified-01	70.9	83.5	79.8	80.9	79.9
Unified-11	74.2	84.1	80.5	80.9	80.7
Unified-00	79.1	87.3	84.0	**83.8**	83.9

Table 6: Local evaluation (f1 scores). The best results that are significantly better than the second best are bold-faced, while those are best but not significantly better than the second best are underlined. All the significance tests are conducted using mention-level McNemar's Chi-square test, with p-value = 0.01.

There are inconsistencies between the entity type names in different datasets, mainly due to different granularities. To remove this unnecessary noise, we manually merged some entity types. For example, we unify Gene and Protein into Gene/Protein as they are commonly used interchangeably; we merge "Simple Chemical" to "Chemical" and leave the problem of entity type granularity for future work. The information in Table 3 and 4 reflects the merged types.

A.1.3 News domain: CoNLL 2003 NER dataset

We use the CoNLL 2003 NER dataset ((Sang and De Meulder, 2003)) to evaluate the models in news domain. More information about the dataset can be found in Table 5. We use synthetic data from the dataset to simulate local training and global evaluation. Specifically, the CoNLL 2003 NER dataset is annotated with four entity types: location, person, organization, and miscellaneous entities. We randomly split the training set into four portions, each contains only one entity type respectively, with other types changed to "O". The models are trained on the four training portions and we test on the original test set with all entity types annotated.

A.1.4 Data split

For the news domain, we use the default train, dev, test portion of the CoNLL 2003 NER dataset. For the biomedicine domain, we follow the data split in Crichton et al. (2017) for both the training and the evaluation groups. All datasets are divided into three portions: train, dev, and test. We train the model on the training set of the training group and tune the hyper-parameters on the corresponding development set. Global evaluations are performed on the test set of the evaluation group.

A.2 Local Evaluation

For a sanity check, we evaluate the models on the training corpora and compare the results with state-of-the-art systems. In this setting, all the models are trained on the training set of the training corpora (without fine-tuning on global evaluation corpora) and evaluated on their test set. The results are shown in Table 6. **STM** is the single-task models we implemented, following the settings in Wang et al. (2018). The SOTA is achieved by Wang et al. (2018) with multi-task model, which is shown in the table as **MTM Wang et al. (2018)**. They trained their model on *BC2GC*, *BC4CHM*, *NCBI*, *JNLPBA*, and *BC5CDR*. **MTM (ours)** is the multi-task model we trained on our five training corpora and used as a baseline in the global evaluations. It has the same architecture as Wang et al. (2018).

As we can see, **MTM Wang et al. (2018)** achieves the best results on 3 out of 4 datasets. And our MTM achieves very similar results, showing it is a strong model on training corpora. Our proposed models do not perform very well when evaluated on the training corpora. But in the global evaluation setting, they perform much better compared to our strong MTM. This demonstrates the superiority of our proposed models on task adaptation.

Learning Dense Representations for Entity Retrieval

Daniel Gillick[*]
Google Research
dgillick@google.com

Sayali Kulkarni[*]
Google Research
sayali@google.com

Larry Lansing[*]
Google Research
llansing@google.com

Alessandro Presta[*]
Google Research
apresta@google.com

Jason Baldridge
Google Research
jasonbaldridge@google.com

Eugene Ie
Google Research
eugeneie@google.com

Diego Garcia-Olano[†]
University of Texas at Austin
diegoolano@gmail.com

Abstract

We show that it is feasible to perform entity linking by training a dual encoder (two-tower) model that encodes mentions and entities in the same dense vector space, where candidate entities are retrieved by approximate nearest neighbor search. Unlike prior work, this setup does not rely on an alias table followed by a re-ranker, and is thus the first fully learned entity retrieval model. We show that our dual encoder, trained using only anchor-text links in Wikipedia, outperforms discrete alias table and BM25 baselines, and is competitive with the best comparable results on the standard TACKBP-2010 dataset. In addition, it can retrieve candidates extremely fast, and generalizes well to a new dataset derived from Wikinews. On the modeling side, we demonstrate the dramatic value of an unsupervised negative mining algorithm for this task.

1 Introduction

A critical part of understanding natural language is connecting specific textual references to real world entities. In text processing systems, this is the task of *entity resolution*: given a document where certain spans of text have been recognized as *mentions* referring to entities, the goal is to link them to unique entries in a knowledge base (KB), making use of textual context around the mentions as well as information about the entities. (We use the term *mention* to refer to the target span along with its context in the document.)

Real world knowledge bases are large (e.g., English Wikipedia has 5.7M articles), so existing work in entity resolution follows a two-stage approach: a first component nominates candidate entities for a given mention and a second one selects the most likely entity among those candidates. This parallels typical *information retrieval* systems that consist of an index and a re-ranking model. In entity resolution, the index is a table mapping *aliases* (possible names) to entities. Such tables need to be built ahead of time and are typically subject to arbitrary, hard cutoffs, such as only including the thirty most popular entities associated with a particular mention. We show that this configuration can be replaced with a more robust model that represents both entities and mentions in the same vector space. Such a model allows candidate entities to be directly and efficiently retrieved for a mention, using nearest neighbor search.

To see why a retrieval approach is desirable, we need to consider how alias tables are employed in entity resolution systems. In the following example, *Costa* refers to footballer *Jorge Costa*, but the entities associated with that alias in existing Wikipedia text are *Costa Coffee*, *Paul Costa Jr*, *Costa Cruises*, and many others—while excluding the true entity.

> ***Costa** has not played since being struck by the AC Milan forward...*

The alias table could be expanded so that last-name aliases are added for all person entities, but it is impossible to come up with rules covering all scenarios. Consider this harder example:

> *...warned Franco Giordano, secretary of the **Refoundation Communists** following a coalition meeting late Wednesday...*

It takes more sophistication to connect the colloquial expression *Refoundation Communists* to the *Communist Refoundation Party*. Alias tables cannot capture all ways of referring to entities in general, which limits recall.

Alias tables also cannot make systematic use of context. In the *Costa* example, the context (e.g., *AC Milan forward, played*) is necessary to know that this mention does not refer to a company or a psychologist. An alias table is blind to this information and must rely only on prior probabilities of entities given mention spans to manage ambiguity. Even if the correct entity is retrieved, it might have such a low prior that the re-ranking model cannot recover it. A retrieval system with access to both the mention span and its context can significantly improve recall. Furthermore, by pushing the work of the alias table into the model, we avoid manual processing and heuristics required for matching mentions to entities, which are often quite different for each new domain.

This work includes the following contributions:

[*]Equal Contributions
[†]Work done during internship with Google

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 528–537
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

- We define a novel dual encoder architecture for learning entity and mention encodings suitable for retrieval. A key feature of the architecture is that it employs a modular hierarchy of sub-encoders that capture different aspects of mentions and entities.
- We describe a simple, fully unsupervised *hard negative mining* strategy that produces massive gains in retrieval performance, compared to using only random negatives.
- We show that approximate nearest neighbor search using the learned representations can yield high quality candidate entities very efficiently.
- Our model significantly outperforms discrete retrieval baselines like an alias table or BM25, and gives results competitive with the best reported accuracy on the standard TACKBP-2010 dataset.
- We provide a qualitative analysis showing that the model integrates contextual information and world knowledge even while simultaneously managing mention-to-title similarity.

We acknowledge that most of the components of our work are not novel in and of themselves. Dual encoder architectures have a long history (Bromley et al., 1994; Chopra et al., 2005; Yih et al., 2011), including for retrieval (Gillick et al., 2018). Negative sampling strategies have been employed for many models and applications, e.g. Shrivastava et al. (2016). Approximate nearest neighbor search is its own sub-field of study (Andoni and Indyk, 2008). Nevertheless, to our knowledge, our work is the first combination of these ideas for entity linking. As a result, we demonstrate the first accurate, robust, and highly efficient system that is actually a viable substitute for standard, more cumbersome two-stage retrieval and re-ranking systems. In contrast with existing literature, which reports multiple seconds to resolve a single mention, we can provide strong retrieval performance across all 5.7 million Wikipedia entities in around 3ms per mention.

2 Related work

Most recent work on entity resolution has focused on training neural network models for the candidate re-ranking stage (Francis-Landau et al., 2016; Eshel et al., 2017; Yamada et al., 2017a; Gupta et al., 2017; Sil et al., 2018). In general, this work explores useful context features and novel architectures for combining mention-side and entity-side features. Extensions include joint resolution over all entities in a document (Ratinov et al., 2011; Globerson et al., 2016; Ganea and Hofmann, 2017), joint modeling with related tasks like textual similarity (Yamada et al., 2017b; Barrena et al., 2018) and cross-lingual modeling (Sil et al., 2018), for example.

By contrast, since we are using a two-tower or dual encoder architecture (Gillick et al., 2018; Serban et al., 2018), our model cannot use any kind of attention over both mentions and entities at once, nor feature-wise comparisons as done by Francis-Landau et al. (2016). This is a fairly severe constraint – for example, we cannot directly compare the mention span to the entity title – but it permits retrieval with nearest neighbor search for the *entire* context against a single, all encompassing representation for each entity.

3 Data

As a whole, the entity linking research space is fairly fragmented, including many task variants that make fair comparisons difficult. Some tasks include named entity recognition (mention span prediction) as well as entity disambiguation, while others are concerned only with disambiguation (the former is often referred to as *end-to-end*. Some tasks include the problem of predicting a *NIL* label for mentions that do not correspond to any entity in the KB, while others ignore such cases. Still other tasks focus on *named* or proper noun mentions, while others include disambiguation of concepts. These variations and the resulting fragmentation of evaluation is discussed at length by Ling et al. (2015) and Hachey et al. (2013), and partially addressed by attempts to consolidate datasets (Cornolti et al., 2013) and metrics (Usbeck et al., 2015).

Since our primary goal is to demonstrate the viability of our unified modeling approach for entity retrieval, we choose to focus on just the disambiguation task, ignoring NIL mentions, where our set of entity candidates includes every entry in the English Wikipedia.

In addition, some tasks include relevant training data, which allows a model trained on Wikipedia (for example) to be tuned to the target domain. We save this fine-tuning for future work.

Training data Wikipedia is an ideal resource for training entity resolution systems because many mentions are resolved via internal hyperlinks (the mention span is the anchor text). We use the 2018-10-22 English Wikipedia dump, which includes 5.7M entities and 112.7M linked mentions (labeled examples). We partition this dataset into 99.9% for training and the remainder for model selection.

Since Wikipedia is a constantly growing an evolving resource, the particular version used can significantly impact entity linking results. For example, when the TACKBP-2010 evaluation dataset was published, Wikipedia included around 3M entities, so the number of retrieval candidates has increased by nearly two times. While this does mean new contexts are seen for many entities, it also means that retrieval gets more difficult over time. This is another factor that makes fair comparisons challenging.

Evaluation data There are a number of annotated datasets available for evaluating entity linking systems. Given the choices discussed above, the **TACKBP-2010**

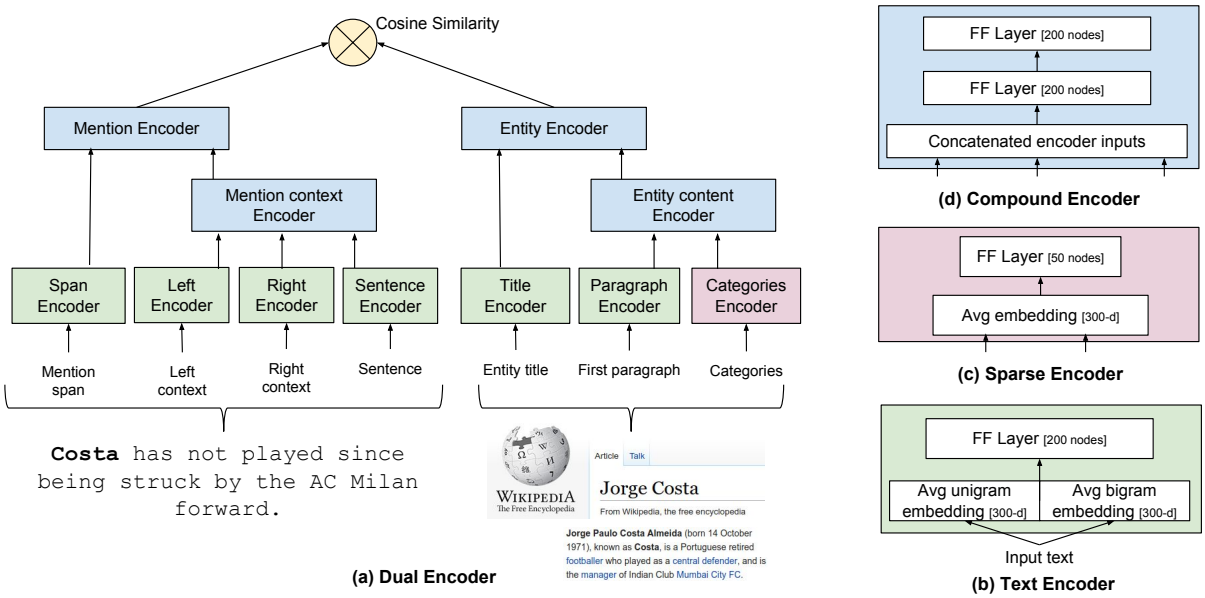


Figure 1: Architecture of the dual encoder model for retrieval (a). Common component architectures are shown for (b) text input, (c) sparse ID input, and (d) compound input joining multiple encoder outputs. Note that all text encoders share a common set of embeddings.

dataset[1] is the most widely used evaluation that matches our constraints and allows us to compare to a reasonable variety of prior work. It includes 1020 annotated mention/entity pairs derived from 1013 original news and web documents. While there is a related development set associated with this evaluation set, we do not use it for any fine-tuning, as explained above.

To further validate our results, we also include a new evaluation set called **Wikinews**, which includes news pages from Wikinews[2] in English for the year 2018. It includes 2263 annotated mention/entity pairs derived from 1801 original documents. Because we pulled these documents at the same time as the Wikipedia dump, the entity annotations are consistent with our training set and have not been subject to the kind of gradual rot that befalls older evaluation data as the updated KB diverges from the annotations. This data is available here: https://github.com/google-research/google-research/tree/master/dense_representations_for_entity_retrieval/.

4 Entity retrieval model

We use nearest neighbor search to retrieve entities based on a mention in context, after learning dense, fixed-length vector representations of each.

4.1 Dual Encoder model

The *dual encoder* is a two-tower architecture suitable for retrieval (Gillick et al., 2018). It has one network structure for encoding mentions (including their contexts), a second for encoding entities (including KB features), and a cosine function to compute similarity between representations (which must be the same dimension). A key property of this architecture is that there is no direct interaction between the encoders on each side. This enables efficient retrieval, but constrains the set of allowable network structures. The dual encoder learns a mention encoder ϕ and an entity encoder ψ, where the score of a mention-entity pair (m, e) defined as:

$$s(m, e) = \cos(\phi(m), \psi(e)) \quad (1)$$

Figure 1 shows the full model architecture and the feature inputs to each encoder. We use a *compound encoder* (Figure 1d) to add useful sub-structure to each tower. The mention-side encoder first combines the context features, and then combines the result with the mention span encoding. Similarly, the entity-side encoder first combines the entity paragraph and categories, and then combines the result with the entity title encoding.

The mention encoder uses four text features to capture both the span text and the textual context surrounding it. The context features include the five tokens immediately to the left and right of the span. In the sentence feature, the mention span is replaced by a special symbol.

The entity encoder uses the entity's title and the first paragraph of its Wikipedia page as text features. It additionally incorporates the unedited user-specified categories associated with the entity. We do not use the entity IDs as features so that the model generalizes more easily to new entities unseen at training time. In fact, more than 1M candidate entities available at retrieval time have no associated training examples, but this architecture allows these to be encoded using their feature representations.

[1] https://tac.nist.gov/
[2] https://en.wikinews.org

A shared embedding look-up is used for all text features (Figure 1b). Specifically, we embed all unigrams and bigrams to get 300-dimensional averaged unigram embeddings and 300-dimensional averaged bigram embeddings for each text feature. Unigram embeddings are initialized from GloVe vectors (Pennington et al., 2014), and we use 5M hash buckets for out-of-vocabulary unigrams and bigrams(Ganchev and Dredze, 2008). These averaged embeddings are concatenated and then passed through a feed-forward layer. For the category features, each entity category name is treated as a sparse input, and the embeddings for all categories for an entity are averaged to produce a 300-dimensional representation, which in turn is passed through a feed-forward layer (Figure 1c).

Our experiments show that this architecture is highly effective for both retrieval and resolution. Nevertheless, we expect that additional modeling ideas will further improve performance, especially for resolution. Recent work such as Durrett and Klein (2014) has shown improvements derived from better, longer-range, context features; similarly, there are many more potentially useful KB-derived features. More complex encoder architectures that use some form of attention over the input tokens and features could also be beneficial.

4.2 Training

Our training data is described in Section 3. The inputs to the entity encoder are constructed from the true entity referred by the landing page. The inputs to the mention encoder are constructed from the source page, using the mention span and surrounding context.

These pairs constitute only positive examples, so we use *in-batch random negatives* (Henderson et al., 2017; Gillick et al., 2018): for each mention-entity pair in a training batch, the other entities in the batch are used as negatives. Computationally, this amounts to building the all-pairs similarity matrix for all mentions and entities in a batch. We optimize softmax loss on each row of the matrix, so that the model is trained to maximize the score of the correct entity with respect to random entities. This is a version of the sampled softmax (Jozefowicz et al., 2016), which we use in place of the full softmax because the normalization term is intractable to compute over all 5.7M entities.

The softmax loss is not directly applied to the raw cosine similarities. Instead, a scalar multiplier a is learned to map the similarities (in the range $[-1, 1]$) to unbounded *logits*. For each training pair (m_i, e_i) in a batch of B pairs, the loss is computed as:

$$L(m_i, e_i) = -f(m_i, e_i) + \log \sum_{j=1}^{B} \exp(f(m_i, e_j)) \tag{2}$$

where

$$f(m_i, e_j) = a \cdot s(m_i, e_j) \tag{3}$$

We track in-batch recall@1 (accuracy) on the held out set during training. For each instance, the model gets a score of 1 if the correct entity is ranked above all in-batch random negatives, 0 otherwise. We stop training after the metric flattens out (about 40M steps).

For all experiments, we use a batch size of 100, standard SGD with Momentum of 0.9 and a fixed learning rate 0.01.

Our aim here is to demonstrate a pure retrieval system, so we train our models solely from Wikipedia and refrain from tuning them explicitly on in-domain documents from the evaluation tasks.

4.3 Hard negative mining

Random negatives alone are not enough to train an accurate entity resolution model because scoring the correct entity above random alternatives can typically be achieved just by comparing the mention text and entity title. More challenging negative examples must be introduced to force the model to exploit context. This strategy is somewhat akin to Importance Sampling (Bengio et al., 2003), for example.

After learning an initial model using random negatives, we identify hard negatives via the following steps:

1. Encode all mentions and entities found in training pairs using the current model.
2. For each mention, retrieve the most similar 10 entities (i.e., its nearest neighbors).
3. Select all entities that are ranked above the correct one for the mention as negative examples.

This yields new negative mention/entity pairs for which the model assigns a high score. It crucially relies on the fact that there is just one correct entity, unlike other tasks that consider general similarity or relatedness (and which are well served by random negatives). For example, negatives mined in this way for paraphrasing or image captioning tasks could actually turn out to be positives that were not explicitly labeled in the data. It is precisely because the distribution over candidate entities that match a contextualized mention tends to have such low entropy that makes negative mining such a good fit for this task.

After merging these with the original positive pairs to construct a classification task, we resume training the initial dual encoder model using logistic loss on this new set of pairs. To retain good performance on random negatives, the new task is mixed with the original softmax task in a multi-task learning setup in which the two loss functions are combined with equal weight and optimized together. For a pair (m, e) with label $y \in \{0, 1\}$, the hard negative loss is defined as:

$$\begin{aligned} L_h(m, e; y) = &- y \cdot \log f(m, e) \\ &- (1 - y) \cdot \log(1 - f(m, e)) \end{aligned} \tag{4}$$

where

$$f(m, e) = g(a_h \cdot s(m, e) + b_h) \tag{5}$$

Here, $g(x) = 1/(1 + e^{-x})$ is the logistic function, and a_h and b_h are learned scalar parameters to transform the cosine similarity into a logit.[3]

For the hard negatives task, we track *Area Under the ROC curve* (AUC) on a held out set of pairs. We stop training when the average of the evaluation metrics for the two tasks stabilizes (about 5M steps).

Finally, we iteratively apply this negative mining procedure. For each round, we mine negatives from the current model as described above and then append the new hard examples to the classification task. Thus each subsequent round of negative mining adds fewer and fewer new examples, which yields a stable and naturally convergent process. As we show in our experiments, iterative hard negative mining produces large performance gains.

4.4 Inference

Once the model is trained, we use the entity encoder to pre-compute encodings for all candidate entities (including those that do not occur in training). At prediction time, mentions are encoded by the mention encoder and entities are retrieved based on their cosine similarity. Since our focus is on model training, we use brute-force search in our evaluation. However, for online settings and larger knowledge bases, an approximate search algorithm is required. In Section 5.2 we show that, when using approximate search, the system retains its strong performance while obtaining a nearly 100x speedup on an already fast retrieval.

5 Experiments

5.1 Evaluation setup

We demonstrate our model performance as compared to a baseline alias table. As is standard, it is built by counting all (mention span, entity) pairs in the training data. The counts are used to estimate prior probabilities $P(e|m)$ of an entity given a mention span (alias); for each entity, the aliases are ordered according to these priors and limited to the top 100 candidates.

$$P(e|m) = \frac{count(e, m)}{\sum_{e' \in E} count(e', m)} \quad (6)$$

Here, $count(e, m)$ is the number of occurrences of mention m linked to entity e, and E is the set of all entities appearing in the data.

Since alias table construction is often extended with various heuristics, we also include a variant that includes unigrams and bigrams of the mention text as additional aliases. This can help when the entities (specifically person names) are referenced as last/first name at inference time.

Finally, since we are primarily concerned with demonstrating performance of a retrieval system (as opposed to a re-ranking system) or a combination of the two, we include results using the standard BM25 retrieval algorithm (the *Gensim* implementation[4]). We found that indexing each entity using its title gave much better results than indexing with the first paragraph text (or the full document text).

We measure recall@k (R@k), defined as the proportion of instances where the true entity is in the top k retrieved items. We report R@1 (accuracy of the top retrieved result), which is standard for TAC/KBP-2010, as well R@100, which better captures overall retrieval performance.

We refer to the models with these abbreviations:

- **AT-Prior**: The alias table ordered by $P(e|m)$.
- **AT-Ext**: The heuristically extended alias table.
- **BM25**: The BM25 retrieval algorithm, where each entity is indexed using its title.
- **DEER**: Our Dual Encoder for Entity Resolution, as described in section 4.

System	R@1	Entities
AT-Prior	71.9	5.7M
AT-Ext	73.3	5.7M
Chisholm and Hachey (2015)	80.7	800K
He et al. (2013)	81.0	1.5M
Sun et al. (2015)	83.9	818K
Yamada et al. (2016)	85.2	5.0M
Nie et al. (2018)	86.4	5.0M
Barrena et al. (2018)	87.3	523K
DEER (this work)	87.0	5.7M

Table 1: Comparison of relevant TACKBP-2010 results using Recall@1 (accuracy). While we cannot control the candidate entity set sizes, we attempt to approximate them here.

5.2 Results

Table 1 provides a comparison against the most relevant related work. While there are some reported improvements due to collective (*global*) resolution of all mentions in a document (Globerson et al. (2016) report 87.2% and Nie et al. (2018) report 89.1%), we limit comparisons to *local* resolution. We also limit comparisons to systems that ignore NIL mentions (referred to as *in-KB* accuracy), so all those reported in the table evaluate precisely the same set of mentions. As noted earlier, we cannot control the candidate sets used in each of these experiments, and we are at some disadvantage given our larger set of candidates.

Retrieval performance Table 2 provides the percent of mentions for which the correct entity is found in the top 100 retrieved results, using the different baselines

[3] The additive parameter is only needed for the logistic loss component, as the softmax function is invariant to translation.

[4] https://radimrehurek.com/gensim/summarization/bm25.html

System	TACKBP-2010	Wikinews
AT-Prior	89.5	93.8
AT-Ext	91.7	94.0
BM25	68.9	83.2
DEER	**96.3**	**97.9**

Table 2: Retrieval evaluation comparison for TACKBP-2010 and Wikinews using Recall@100.

Method	Mean search time (ms)	Wikinews R@100
Brute force	291.9	97.88
AH	22.6	97.22
AH+Tree	3.3	94.73

Table 3: Comparison of nearest-neighbor search methods using the DEER model. The benchmark was conducted on a single machine. AH indicates quantization-based asymmetric hashing; AH+Tree adds an initial tree search to further reduce the search space.

and the DEER model. The learned representations deliver superior performance and do not require special handling for unigrams versus bigram lookups, counts for entity prominence, and so on.

Resolution performance To put DEER's architecture and performance in context, we compare it with prior work in some more detail here.

He et al. (2013) use a dual encoder setup to train a re-ranker, but start with unsupervised training to build representations of contexts and entities using Denoising Autoencoders. They use an alias table for candidate generation, and then train a ranking model using mention-specific batching to obtain hard in-batch negatives. Our results suggest that the autoencoder pretraining is not necessary and that our unsupervised negative mining can outperform heuristic selection of negatives.

Sun et al. (2015) also use a dual encoder that has similar structure to ours. Like He et al. (2013), they use it to score entries from an alias table rather than directly for retrieval. Their mention encoder is a considerably more complex combination of mention and context rather than the simple compounding strategy in our architecture. Their alias table method not only maps mentions to entities, but also uses additional filters to reduce the set of candidate entities based on words in the mention's context. They report that this method has a recall of 91.2% on TACKBP 2010, while our direct retrieval setup gives Recall@100 of 96.3% (see Table 2). They train their representations for each true mention-entity pair against a single random negative entity for the mention, whereas our method takes advantage of the entire batch of random negatives as well further refinement through hard negative mining.

Yamada et al. (2016) use an alias table derived from the December 2014 Wikipedia dump, restricted to the fifty most popular entities per mention. They tune their model on the TACKBP 2010 training set. Architecturally, they include features that capture the alias table priors and string similarities, both of which are not feasible in a dual encoder configuration that precludes direct comparison between mention- and entity-side features. DEER's better results indicate that learned representations of mentions and entities can be powerful enough for entity retrieval even without any cross-attention.

Nie et al. (2018) define a complex model that uses both entity type information and attention between the mention string and the entity description. To augment the small 1500 example training data in TACKBP, they also collected 55k mentions found in Wikipedia that were active in TACKBP 2010 to train this model. DEER is simply trained over all entities in Wikipedia and uses no cross-attention or explicit type information, yet delivers better resolution performance.

Most standard entity linking models build a single ranking model on top of the candidate set provided by an alias table. Barrena et al. (2018) instead train 523k mention-specific deep classifiers—effectively treating entity linking as a special form of word sense disambiguation. They do this by pre-training a single LSTM that predicts among 248k mentions, and then the parameters of this model are used to warm start each of the 523k mention-specific models. In doing so, they learn an effective context encoding, and can then fine-tune each mention model to discriminate among the small set of popular candidate entities for the mention (their alias table uses a cutoff of the thirty most popular entities for each mention). DEER in contrast, has a single mention encoder that is simple and fast, and performs nearly equivalently while retrieving from a much larger set of entities.

Approximate search Tables 1 and 2 report performance using brute force nearest-neighbor search. That is, we score each mention against all 5.7M entities to get the top k neighbors. However, a major motivation for using a single-stage retrieval model is that it can allow scaling to much larger knowledge bases by reducing retrieval time via approximate search.

To estimate performance in a real-world setting, we repeat the evaluation of DEER using the quantization-based approaches described by Guo et al. (2016). Table 3 shows the trade-off between search time and recall on Wikinews. Compared to brute force, search time can be reduced by an order of magnitude with a small loss in R@100, or by two orders of magnitude while losing less than 3 points. This is crucial for scaling the approach to even larger KBs and supporting the latency requirements of real-world applications.

Impact of hard negative mining Figure 2 shows the improvement in Recall@1 from each round of hard negative mining. The first iteration gives a large improvement over the initial round of training with only random negatives. Successive iterations yield further gains, eventually flattening out. Our strategy of append-

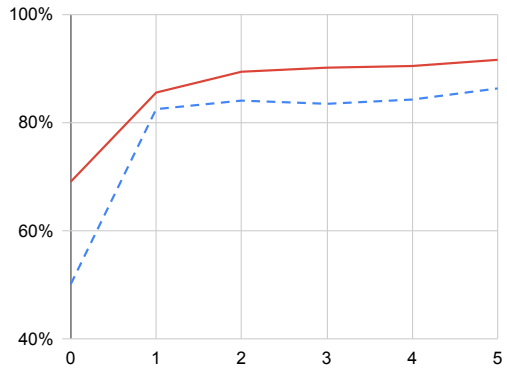


Figure 2: Recall@1 improvement for successive iterations of hard negative mining for Wikinews (solid) and TACKBP-2010 (dashed).

ing each new set of hard negatives to the previously mined ones means that each new set has proportionately less influence—this trades off some opportunity for improving the model in favor of stability.

5.3 Qualitative analysis

Here, we show a variety of examples to elucidate how DEER effectively models context, and to provide intuition for the learned entity representations.

First, we compare entities retrieved by DEER with those retrieved by the alias table baseline. Table 5 shows some instances where the alias table does not contain the correct entity for a given mention text (in the top 100 neighbors) or it fails to return any entity at all. In all of these cases, it is clear that context is essential. While a scoring stage is intended to leverage context, it is limited by the set of retrieved entities; our approach uses context directly.

For example, mentions like *Costa* and *Justin* are linked to the correct entities in the alias table, but with such low prior probability that we would need to retrieve far more than the top 100 entities to consider them. At the other extreme, mentions like *Refoundation Communists* and *European EADS* are missed by the baseline because they don't have direct string matches in the alias table. Additional extensions to our alias table allowing token re-ordering could help catch the former (though this might reduce precision too much), but it's unlikely that any alias table could connect *European EADS* with *Airbus* in the absence of an explicit anchor-text link. This example helps highlight how a fully learned retrieval model can generalize to new data.

Second, Table 6 shows more directly how modifying the context around a mention span changes the retrieved entities. For example, the model correctly differentiates between Phoenix the city, Phoenix the band, and Phoenix in mythology based on the sentence surrounding an identical mention span.

Third, since our model produces encodings for all 5.7M entities, we can retrieve nearest neighbors for any entity. Some examples are shown in Table 4. The model tends to prefer related entities of the same type, and often ones that share portions of their names, probably because entity titles are so important to linking with mentions. The nearest neighbors for *Jorge Costa*, our running example, include a variety of retired Portuguese football players, many of whom have *Costa* in their names.

Entity	Nearest neighbors
Jorge Costa	José Alberto Costa, Eduardo Costa, Peter Shilton, Rui Costa, Nuno Gomes, Ricardo Costa (Portuguese footballer), André Gomes, Bruno Ribeiro, Diego Costa
Costa Cruises	MSC Cruises, P&O Cruises, Princess Cruises, Island Cruises, AIDA Cruises, Silversea Cruises, Carnival Corporation & plc, Costa Concordia, Celebrity Cruises
Arctic sea ice decline	Arctic ice pack, Measurement of sea ice, Arctic geoengineering, Arctic sea ice ecology and history, Climate Change Science Program, Abrupt climate change, Sea ice thickness, Antarctic sea ice, Marine ice sheet instability
Pink Floyd	Led Zeppelin, The Who, Duran Duran, Syd Barrett, The Velvet Underground, Eddie Floyd, The Beatles, The Australian Pink Floyd Show, Roger Waters

Table 4: Nearest neighbors retrieved by DEER for a sample of entities.

Finally, Figure 3 is a t-SNE projection of the entity encodings for a selection of cities, bands, and people (nobel literature winners). The cities and bands were chosen to have high word overlap, e.g. *Montreal (city)* and *Of Montreal (band)*, to demonstrate how our entity embeddings differ from standard word embeddings. Note also the sub-clusters that form within each type cluster. Latin American authors cluster together, as do the Existentialists; the cities have some geographical proximity, though Brazil and Portugal are neighbors, presumably because of shared language and culture.

6 Conclusion

Our results with DEER show that a single-stage retrieval approach for entities from mentions is highly effective: without any domain-specific tuning, it performs at least as well as the best comparable two-stage systems. While our bag-of-ngrams encoders provided a strong proof of concept, we can almost certainly improve results with more sophisticated encoders, using a BERT architecture (Devlin et al., 2019), for example. Further, by virtue of approximate search techniques, it can be used for very

Mention	Baseline predictions	Model predictions
Costa has not played since being struck by the AC Milan forward	Costa Coffee, Paul Costa Jr, Comstock-Needham system, Costa Cruises, Achille Costa	Ricardo Costa (Portuguese footballer), Fernando Torres, Pedro (footballer born 1987), **Jorge Costa**
Australia beat West Indies by five wickets in a **World Series** limited overs match	World Series, ATP International Series, 2010 World Series	**World Series Cricket**, The University Match (cricket), Australian Tri-Series
Justin made his second straight start for Harbaugh, who has a knee injury	Justin (historian), Justin Martyr, Justin (consul 540)	**Paul Justin**, Joe Montana, Dale Steyn
plays for the Cape Town-based **Cobras** franchise	Cobra, AC Cobra, Indian Cobra	Cobra, Snake, **Cape Cobras**
OSI reports profit on overseas cancer drug sales	Open Systems Interconnection, Open Source Initiative, OSI (band)	Health Insurance Portability and Accountability Act, **OSI Pharmaceuticals**, James L. Jones
EVN imposed **rotating power cuts** earlier this year as the worst drought in a century dropped water levels	no matches	Cogeneration, **Power outage**, Scram
warned Franco Giordano, secretary of the **Refoundation Communists** following a coalition meeting late Wednesday	no matches	League of Communists of Yugoslavia, **Communist Refoundation Party**, Communist Party of Spain
The **European EADS** consortium, which makes the Eurofighter Typhoon, said it was not comfortable with the NATO-member countries' bidding process	no matches	**Airbus**, Airbus A400M Atlas, NATO
such as the record California wildfires, high temperature extremes, retreating glaciers, and melting snow cover, the **decline of sea ice**, rising sea levels with increasing ocean acidification and coastal flooding	no matches	Moskstraumen, **Arctic sea ice decline**, Glacier, Sea ice

Table 5: Examples of test mentions that require making use of context, where the alias table does not retrieve the correct entity. We show the top entities returned by both systems, with the correct entity in bold.

Mention	Model predictions
From 1996, **Cobra** was brewed under contract by Charles Wells Ltd and experienced strong growth in sales for the next ten years.	The Cobra Group, **Cobra Beer**, Cobra (Tivoli Friheden)
Guys fondly remembered **Cobra** - the band from Memphis featuring Jimi Jamison and Mandy Meyer who released one Album - Frist Strike - before the Band split!	**Cobra (American band)**, Wadsworth Jarrell, Cobra Records, Cobra (Japanese band)
Since the late 18th century, **Paris** has been famous for its restaurants and haute cuisine, food meticulously prepared and artfully presented.	**Paris**, Nice, Bucharest
Kim Kardashian may be a household name now, but that wasnt always the case - and it may all be because of pal **Paris**.	Paris, **Paris Hilton**, Paris syndrome
Rory and **Paris** are the only two people on Gilmore Girls who share the same goals.	Paris, Paris (mythology), **Paris Geller**
Texas was finally annexed when the expansionist James K. Polk won the election of 1844 who ordered General Zachary Taylor south to the Rio Grande on January 13, 1846.	**Texas**, Texas annexation, Texas in the American Civil War
Fronted by Sharleen Spiteri, **Texas** have released eight studio albums and are known for songs such as 'I Don't Want a Lover', 'Say What You Want', 'Summer Son' and 'Inner Smile'	**Texas (band)**, Texas, Tich (singer)
There is an amazing piece of historic architecture set in downtown **Phoenix** that was build in 1929 in the Spanish Baroque style and features intricate murals and moldings.	**Phoenix, Arizona**, Prescott, Arizona
Phoenix once again played another late night show, now they have Late Night with Jimmy Fallon where they played a great rendition of 'Lisztomania'	**Phoenix (band)**, Joaquin Phoenix, Phoenix, Arizona
According to Greek mythology, the **Phoenix** lived in Arabia next to a well where the Greek sun-god Apollo stopped his chariot in order to listen to its song.	Phoenix (mythology), Phoenix (son of Amyntor), **Phoenix (son of Agenor)**

Table 6: Changing the context around a mention span changes the mention encoding, and thus the set of retrieved neighbors.

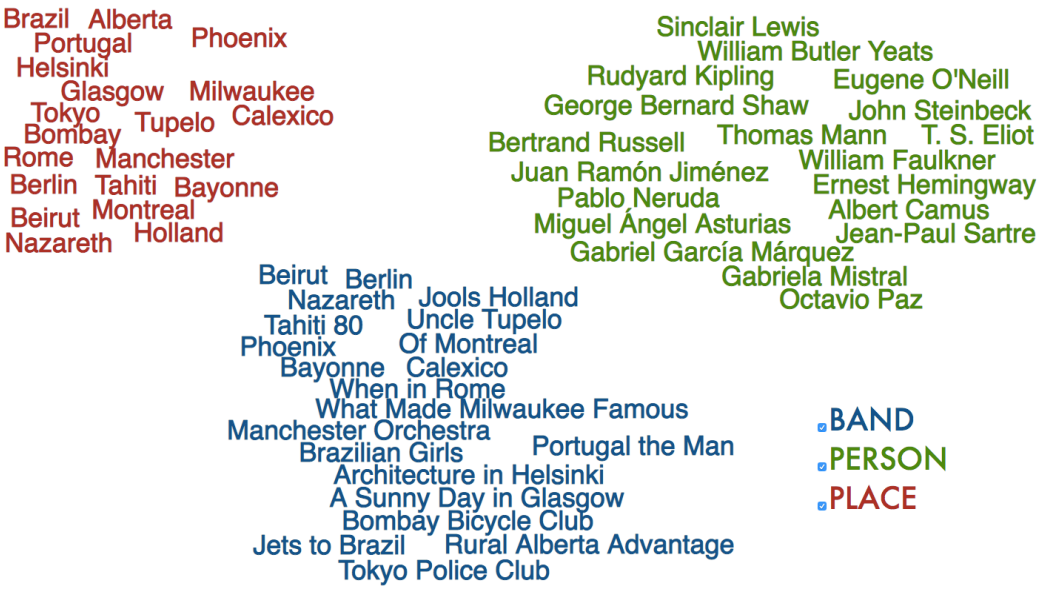


Figure 3: A 2D projection of cities, bands, and people embeddings (using t-SNE), color coded by their category.

fast retrieval, and is likely to scale reasonably to much larger knowledge bases.

We also note that the dual encoder approach allows for interesting extensions beyond traditional entity linking. For example, the context encodings provide a natural model for building entity expectations during text processing, such that entities relevant to the context can be retrieved and used for reference resolution as a document is processed incrementally. We expect this will be useful for collective entity resolution as well as modeling coherence.

Finally, while we focus on training with English Wikipedia, Sil et al. (2018) show that using cross-lingual datasets can help to refine the context information more effectively. Since English constitutes only a fraction of the total Wikipedia, and entity IDs are (mostly) language-independent, there is great opportunity to extend this work to far more training examples across far more languages.

Acknowledgements

The authors would like to thank Ming-Wei Chang, Jan Botha, Slav Petrov, and the anonymous reviewers for their helpful comments.

References

Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM* 51(1):117.

Ander Barrena, Aitor Soroa, and Eneko Agirre. 2018. Learning text representations for 500k classification tasks on named entity disambiguation. In *CoNLL*. pages 171–180.

Yoshua Bengio, Jean-Sébastien Senécal, et al. 2003. Quick training of probabilistic neural nets by importance sampling. In *AISTATS*. pages 1–9.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*. pages 737–744.

Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *Transactions of the ACL* 3:145–156.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*. IEEE, volume 1, pages 539–546.

Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. 2013. A framework for benchmarking entity-annotation systems. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, pages 249–260.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*. pages 4171–4186.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. In *Transactions of the ACL*.

Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. 2017. Named entity disambiguation for noisy text. In *CoNLL*. Vancouver, Canada, pages 58–68.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *NAACL-HLT*. San Diego, California, pages 1256–1261.

Kuzman Ganchev and Mark Dredze. 2008. Small statistical models by random feature mixing. In *Proceedings of the ACL-08: HLT Workshop on Mobile Language Processing*. pages 19–20.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. *CoRR* abs/1704.04920.

Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. 2018. End-to-end retrieval in continuous space .

Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard, and Fernando Pereira. 2016. Collective entity resolution with multi-focal attention. In *ACL*.

Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. 2016. Quantization based fast inner product search. In *Artificial Intelligence and Statistics*. pages 482–490.

Nitish Gupta, Sameer Singh, and Dan Roth. 2017. Entity linking via joint encoding of types, descriptions, and context. In *EMNLP*. pages 2681–2690.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R Curran. 2013. Evaluating entity linking with wikipedia. *Artificial intelligence* 194:130–150.

Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *ACL*. Sofia, Bulgaria, pages 30–34.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yunhsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652* .

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. *CoRR* abs/1708.00781.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* .

Xiao Ling, Sameer Singh, and Daniel S Weld. 2015. Design challenges for entity linking. *Transactions of the ACL* 3:315–328.

Feng Nie, Yunbo Cao, Jinpeng Wang, Chin-Yew Lin, and Rong Pan. 2018. Mention and entity description co-attention for entity disambiguation. In *AAAI*. Vancouver, Canada, pages 5908–5915.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL-HLT*. Stroudsburg, PA, USA, pages 1375–1384.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2018. A survey of available corpora for building data-driven dialogue systems: The journal version. *D&D* 9(1):1–49.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. 2016. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pages 761–769.

Avirup Sil, Gourab Kundu, Radu Florian, and Wael Hamza. 2018. Neural cross-lingual entity linking. In *AAAI*. Vancouver, Canada, pages 5465–5472.

Yaming Sun, Lei Lin, Duyu Tang, Nan Yang, Zhenzhou Ji, and Xiaolong Wang. 2015. Modeling mention, context and entity with neural networks for entity disambiguation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Ricardo Usbeck, Michael Röder, Axel-Cyrille Ngonga Ngomo, Ciro Baron, Andreas Both, Martin Brümmer, Diego Ceccarelli, Marco Cornolti, Didier Cherix, Bernd Eickmann, et al. 2015. In *Proceedings of the 24th international conference on World Wide Web*. pages 1133–1143.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *arXiv preprint arXiv:1601.01343* .

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017a. Learning distributed representations of texts and entities from knowledge base. *TACL* 5:397–411.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017b. Learning distributed representations of texts and entities from knowledge base. *CoRR* abs/1705.02494.

Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *CoNLL*. pages 247–256.

Chenwei Zhang, Yaliang Li, Nan Du, Wei Fan, and Philip S. Yu. 2018. Generative discovery of relational medical entity pairs.

CogniVal: A Framework for Cognitive Word Embedding Evaluation

Nora Hollenstein[1], Antonio de la Torre[1], Nicolas Langer[2], Ce Zhang[1]
[1] Department of Computer Science, ETH Zurich
{noraho,antonide,ce.zhang}@ethz.ch
[2] Department of Psychology, University of Zurich
n.langer@psychologie.uzh.ch

Abstract

An interesting method of evaluating word representations is by how much they reflect the semantic representations in the human brain. However, most, if not all, previous works only focus on small datasets and a single modality. In this paper, we present the first multimodal framework for evaluating English word representations based on cognitive lexical semantics. Six types of word embeddings are evaluated by fitting them to 15 datasets of eye-tracking, EEG and fMRI signals recorded during language processing. To achieve a global score over all evaluation hypotheses, we apply statistical significance testing accounting for the multiple comparisons problem. This framework is easily extensible and available to include other intrinsic and extrinsic evaluation methods. We find strong correlations in the results between cognitive datasets, across recording modalities and to their performance on extrinsic NLP tasks.

1 Introduction

Word embeddings are the corner stones of state-of-the-art NLP models. Distributional representations which interpret words, phrases, and sentences as high-dimensional vectors in semantic space have become increasingly popular. These vectors are obtained by training language models on large corpora to encode contextual information. Each vector represents the meaning of a word.

Evaluating and comparing the quality of different word embeddings is a well-known, largely open challenge. Currently, word embeddings are evaluated with extrinsic or intrinsic methods. Extrinsic evaluation is the process of assessing the quality of the embeddings based on their performance on downstream NLP tasks, such as question answering or entity recognition. However, embeddings can be trained and fine-tuned for specific tasks, but this does not mean that they accurately reflect the meaning of words.

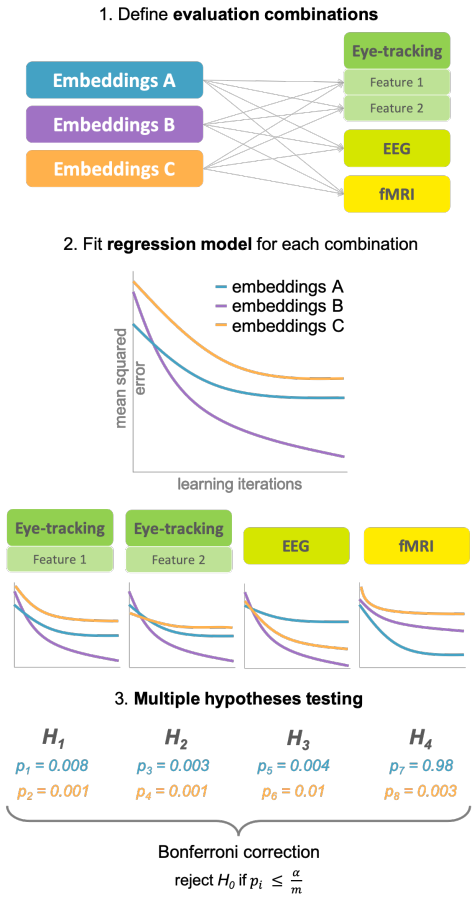


Figure 1: Overview of the cognitive word embedding evaluation process.

One the other hand, intrinsic evaluation methods, such as word similarity and word analogy tasks, merely test single linguistic aspects. These tasks are based on conscious human judgements. Conscious judgements can be biased by subjective factors and the tasks themselves might also be biased (Malvina Nissim, 2019). Additionally, the correlation between intrinsic and extrinsic metrics is not very clear, as intrinsic evaluation results fail

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 538–549
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

to predict extrinsic performance (Chiu et al., 2016; Gladkova and Drozd, 2016). Finally, both intrinsic and extrinsic evaluation types often lack statistical significance testing and do not provide a global quality score.

In this paper, we focus on the *intrinsic subconscious evaluation method* (Bakarov, 2018b), *which evaluates English word embeddings against the lexical representations of words in the human brain, recorded when passively understanding language*. Cognitive lexical semantics proposes that words are defined by how they are organized in the brain (Miller and Fellbaum, 1992). As a result, brain activity data recorded from humans processing language is arguably the most accurate mental lexical representation available (Søgaard, 2016). Recordings of brain activity play a central role in furthering our understanding of how human language works. To accurately encode the semantics of words, we believe that embeddings should reflect this mental lexical representation.

Evaluating word embeddings with cognitive language processing data has been proposed previously. However, the available datasets are not large enough for powerful machine learning models, the recording technologies produce noisy data, and most importantly, only few datasets are publicly available. Furthermore, since brain activity and eye-tracking data contain very noisy signals, correlating distances between representations does not provide sufficient statistical power to compare embedding types (Frank, 2017). For this reason we evaluate the embeddings by exploring how well they can predict human processing data. We build on Søgaard (2016)'s theory of evaluating embeddings with this task-independent approach based on cognitive lexical semantics and examine its effectiveness. The design of our framework follows three principles:

1. **Multi-modality**: Evaluate against various modalities of recording human signals to counteract the noisiness of the data.
2. **Diversity** within modalities: Evaluate against different datasets within one modality to make sure the number of samples is as large as possible.
3. **Correlation** of results should be evident across modalities and even between datasets of the same modality.

Contributions We present CogniVal, the first framework of cognitive word embedding evaluation to follow these principles and analyze the findings. We evaluate different embedding types against a combination of 15 cognitive data sources, acquired via three modalities: eye-tracking, electroencephalography (EEG) and functional magnetic resonance imaging (fMRI). The word representations are evaluated by assessing their ability of predicting cognitive language processing data. After fitting a neural regression model for each combination, we apply multiple hypotheses testing to measure the statistical significance of the results, taking into account multiple comparisons (see Figure 1). This contributes to the consistency of the results and to attain a global score of embedding quality. Our main findings when evaluating six state-of-the-art word embeddings with CogniVal show that the majority of embedding types significantly outperform a baseline of random embeddings when predicting a wide range of cognitive features. Additionally, the results show consistent correlations between between datasets of the same modality and across different modalities, validating the intuition of our approach. Finally, we present an exploratory but promising correlation analysis between the scores obtained using our intrinsic evaluation methods and the performance on extrinsic NLP tasks.

The code of this evaluation framework is openly available[1]. It can be used as is, or in combination with other intrinsic as well as extrinsic evaluation methods for word representations.

2 Related Work

Mitchell et al. (2008) pioneered the use of word embeddings to predict patterns of neural activation when subjects are exposed to isolated word stimuli. More recently, this dataset and other fMRI resources have been used to evaluate learned word representations.

For instance, Abnar et al. (2018) and Rodrigues et al. (2018) evaluate different embeddings by predicting the neuronal activity from the 60 nouns presented by Mitchell et al. (2008). Søgaard (2016) shows preliminary results in evaluating embeddings against continuous text stimuli in eye-tracking and fMRI data. Moreover, Beinborn et al. (2019) recently presented an extensive set of language–brain encoding experiments. Specifically, they evaluated the ability of an ELMo language model to predict brain responses of multiple

[1] https://github.com/DS3Lab/cognival

fMRI datasets.

EEG data has been used for similar purposes. Schwartz and Mitchell (2019) and Ettinger et al. (2016) show that components of event-related potentials can successfully be predicted with neural network models and word embeddings.

However, these approaches mostly focus on one modality of brain activity data from small individual cognitive datasets. The lack of data sources has been one reason why this type of evaluation has not been too popular until now (Bakarov, 2018a). Hence, in this work we collected a wide range of cognitive data sources ranging from eye-tracking to EEG and fMRI to ensure coverage of different features, and consequently of the cognitive processing taking place in the human brain during reading.

Evidence from cognitive neuroscience Murphy et al. (2018) review computational approaches to the study of language with neuroimaging data and show how different type of words activate neurons in different brain regions. Similarly, mapping fMRI data from subjects listening to stories to the activated brain regions, revealed semantic maps of how words are distributed across the human cerebral cortex (Huth et al., 2016).

Furthermore, word predictability and semantic similarity show distinct patterns of brain activity during language comprehension: semantic distances can have neurally distinguishable effects during language comprehension (Frank and Willems, 2017). These findings support the theory that brain activity data does reflect lexical semantics and is thus an appropriate foundation for evaluating the quality of word embeddings.

3 Word embeddings

Pre-trained word vectors are an essential component in state-of-the-art NLP systems. We chose six commonly used pre-trained embeddings to evaluate against the cognitive data sources. See Table 1 for an overview of the dimensions of each embedding type. We evaluate the following types of word embeddings:

- **Glove**: Pennington et al. (2014) provide embeddings of different dimensions trained on aggregated global word-word co-occurrence statistics over a corpus of 6 billion words.

embeddings	dim.	hidden layer units
Glove	50	[30, 26, 20, 5]
Glove	100	[50, 30]
Glove	200	[100, 50]
Glove	300	[150, 50]
Word2vec	300	[150, 50]
WordNet2vec	850	[400, 200]
FastText	300	[150, 50]
ELMo	1024	[600, 200]
BERT	768	[400, 200]
BERT	1024	[600, 200]

Table 1: Overview of word embeddings evaluated with CogniVal. The last column shows the search space of the grid search for the number of units in the hidden layer.

- **Word2vec**: Non-contextual embeddings trained on 100 billion words from a Google News dataset (Mikolov et al., 2013).

- **WordNet2Vec** (Saedi et al., 2018) These embeddings represent the conversion from semantic networks into semantic spaces. Trained on WordNet, a lexical ontology for English that comprises over 155,000 lemmas (but trained only on 60,000 words).

- **FastText** pre-trained embeddings use character n-grams to compose the vector of the full words (Mikolov et al., 2018). We evaluate the embeddings with and without subword information trained on 16 billion tokens of Wikipedia sentences as well as the ones trained on 600 billion tokens of Common Crawl.

- **ELMo** models both complex characteristics of word use (i.e. syntax and semantics), and how these uses vary across linguistic contexts (Peters et al., 2018). These word vectors are learned functions of the internal states of a deep bidirectional language model, which is pre-trained on a large text corpus. We take the first of the three output layers, containing the context insensitive word representations.

- **BERT** embeddings are contextual, bidirectional word representations, based on the idea that fine-tuning a pre-trained language model can help the model achieve better results in the downstream tasks (Devlin et al., 2019). We take the hidden states of the second to last

of 12 output layers as the representation for each token.

4 Cognitive data

In this paper, we consider three modalities of recording cognitive language processing signals: eye-tracking, electroencephalography (EEG), and functional magnetic resonance imaging (fMRI). All three methods are complementary in terms of temporal and spatial resolution as well as the directness in the measurement of neural activity (Mulert, 2013). For the word embedding evaluation we selected a wide range of datasets from these three modalities to ensure a more diverse and accurate representation of the brain activity during language processing.

Table 2 shows an overview of the cognitive data sources used, which are described in more detail below. Since the processing in the brain differs depending on whether the information is accessed via the visual or auditory system (Price, 2012), we include data of different stimuli, e.g. participants reading sentences or listening to audiobooks. Moreover, our collection of cognitive data sources contains datasets of both isolated (single words) and continuous (words in context, i.e. sentences or stories) stimuli. All datasets include English language stimuli and the participants were native speakers or highly proficient.

Eye-tracking Eye-tracking is an indirect measure of cognitive activity. Gaze patterns are highly correlated with the cognitive load associated with different stages of human text processing (Rayner, 1998). For instance, fixation duration is higher for long, infrequent and unfamiliar words (Just and Carpenter, 1980).

All eye-tracking datasets used in this work were recorded from natural, self-paced reading. Each dataset provides different eye-tracking features. The most common features, available in all 7 datasets are: first fixation duration, first pass duration, mean fixation duration, total fixation duration and number of fixations. For a complete list and description of the eye-tracking features available in each corpus see Appendix A.1.

Gaze vectors consist of specific features, which are extracted based on the reading times, fixations and regressions on each word. Feature values are aggregated on word type level and scaled between 0 and 1. The feature values were averaged over all subjects within a dataset. This preprocessing step is done separately for each data source before combining them. Hollenstein and Zhang (2019) show that combining gaze data from different sources can be helpful for NLP applications, even when they are recorded with different devices and filtering,

By using as many features as available from each dataset, including features characterizing basic, early and late word processing aspects, the goal is to cover the whole language understanding process on word level.

EEG Electroencephalography records electrical activity from the brain. It measures voltage fluctuations through the scalp with high temporal resolution.oh (Hauk and Pulvermüller, 2004) presents evidence for the modulation of early electrophysiological brain responses by word frequency. This is evidence that lexical access from written word stimuli is an early process that follows stimulus presentation by less than 200 ms.

The EEG datasets used in this work were either recorded from reading sentences or listening to natural speech. Word-level brain activity could be extracted by mapping to eye-tracking cues (ZuCo), by mapping to auditory triggers (Natural Speech), by recording only the last word in each sentence (N400), or through serial presentation of the words (UCL). Standard preprocessing steps for EEG data, including band-pass filtering and artifact removal, are performed in the same manner for all four data sources. See Appendix A.2 for details on EEG preprocessing.

The EEG data is aggregated over all available subjects and over all occurrences of a token. This yields an n-dimensional vector, where n is the number of electrodes, ranging from 32 to 130, depending on the EEG device used to record the data.

EEG data can be aggregated over all subjects within one dataset, because the number and locations of electrodes are identical. However, due to the differences in the number of electrodes between datasets, we cannot aggregate over all EEG datasets.

fMRI Functional magnetic resonance imaging is a technique for measuring and mapping brain activity by detecting changes associated with blood flow. fMRI has a temporal resolution of two seconds, which means that with continuous stimuli such as natural reading or story listening, one scan covers multiple words. We use datasets of

	Data source	stimulus	subj.	tokens	types	coverage
EYE-TRACKING	GECO (Cop et al., 2017)	text	14	68606	5383	95%
	DUNDEE (Kennedy et al., 2003)	text	10	58598	9131	94%
	CFILT-SARCASM (Mishra et al., 2016)	text	5	23466	4237	85%
	ZUCO (Hollenstein et al., 2018)	text	12	13717	4384	90%
	CFILT-SCANPATH (Mishra et al., 2017)	text	5	3677	1314	89%
	PROVO (Luke and Christianson, 2017)	text	84	2743	1192	95%
	UCL (Frank et al., 2013)	text	43	1886	711	98%
	ALL EYE-TRACKING (aggregated)	text	-	26353	16419	88%
EEG	ZuCo (Hollenstein et al., 2018)	text	12	13717	4384	90%
	NATURAL SPEECH (Broderick et al., 2018)	speech	19	12000	1625	98%
	UCL (Frank et al., 2015)	text	24	1931	711	98%
	N400 (Broderick et al., 2018)	text	9	150	140	100%
fMRI	HARRY POTTER (Wehbe et al., 2014)	text	8	5169	1295	92%
	ALICE (Brennan et al., 2016)	speech	27	2066	588	99%
	PEREIRA (Pereira et al., 2018)	text/image	15	180	180	99%
	NOUNS (Mitchell et al., 2008)	image	9	60	60	100%

Table 2: Cognitive data sources used in this work. Coverage is the percentage of vocabulary in data source occurs in British National Corpus list of most frequent English words[2].

isolated stimuli (e.g the NOUNS dataset) as well as continuous stimuli (e.g. HARRY POTTER). While it is easier to extract word-level signals from isolated stimuli, continuous stimuli allow extracting signals in context over a wider vocabulary.

Where multiple trials were available, the brain activation for each word is calculated by taking the mean over the scans. Moreover, if the stimulus is continuous (HARRY POTTER and ALICE datasets), the data is aligned with an offset of four seconds to account for hemodynamic delay[3].

fMRI data contains representations of neural activity of millimeter-sized cubes called voxels. Standard fMRI preprocessing methods such as motion correction, slice timing correction and co-registration had already been applied before. To select the voxels to be predicted we use the pipeline provided by Beinborn et al. (2019). This pipeline consists of extracting corresponding scan(s) for each word, and randomly selecting 100, 500 and 1000 voxels (for the HARRY POTTER, PEREIRA and NOUNS datasets). The published version of the ALICE dataset provided the preprocessed signal averaged for six regions of interest, hence for this particular dataset we predict the activation for these regions only. Appendix A.3 contains the details of the preprocessing steps. Finally, the fMRI data is converted to n-dimensional vectors, where n is the number of randomly selected voxels (100, 500 or 1000) or regions (6).

5 Embedding evaluation method

In order to evaluate the word embeddings against human lexical representations, we fit the embeddings to a wide range cognitive features, i.e. eye-tracking features and activation levels of EEG and fMRI. This section describes how these models were trained and evaluated. After evaluating each combination separately, we test for statistical significance taking into account the multiple comparisons problem. See Figure 1 for an overview of the evaluation process.

5.1 Models

We fit neural regression models to map word embeddings to cognitive data sources. Predicting multiple features from different sources and modalities allows us to evaluate different aspects of capturing the semantics of a word. Hence, separate models are trained for all combinations. For instance, fitting FastText embeddings to EEG vectors from ZUCO, or fitting ELMo embeddings to

[2] https://www.kilgarriff.co.uk/bnc-readme.html

[3] The fMRI signal measures a brain response to a stimulus with a delay of a few seconds, and it decays slowly over a duration of several seconds (Miezin et al., 2000). For continuous stimuli, this means that the response to previous stimuli will have an influence on the current signal. Thus, context of the previous words is taken into account

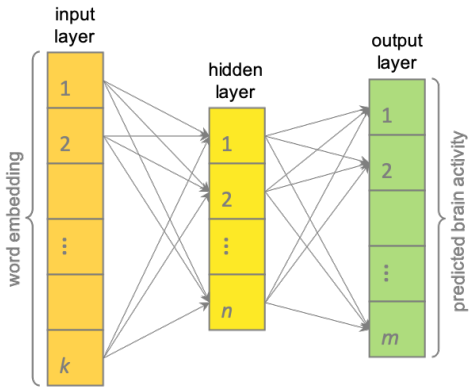


Figure 2: Neural architecture of regression models.

first fixation durations of the DUNDEE corpus.

For the regression models, we train neural networks with k input dimensions, one dense hidden layer of n nodes using ReLU activation and an output layer of m nodes using linear activation. The model is a multiple regression with layers of dimension k-n-m, where k is the number of dimensions of the word embeddings and m changes depending on the cognitive data source to be predicted. For predicting single eye-tracking features m equals 1, whereas for predicting EEG of fMRI vectors m is the dimension of the cognitive data vector, or more specifically, the number of electrodes in the EEG data or the number of voxels in the fMRI data. Figure 2 shows this neural architecture. The loss function optimizes the mean squared error (MSE) and uses an Adam optimizer with a learning rate of 0.001.

5-fold cross validation is performed for each model (80% training data and 20% test data). The optimal number of nodes n in the hidden layer is selected individually for each combination of cognitive data source and embedding type. To this end, a grid search is performed before training, which is evaluated on a validation set consisting of 20% of the training data with 3-fold cross validation (see Table 1 for details on the search space). The best model is then saved and used to predict the cognitive feature for each word in the test set. Finally, the results are measured with the mean squared error, averaged over all predicted words.

CogniVal allows for evaluation against another word embedding type as well as evaluation against a random baseline. To generate a fair baseline we create random vectors for each word of n dimensions, corresponding to the same number of dimensions of the embeddings to be evaluated.

	voxels		
embeddings	100	500	1000
glove-300	0.119	0.081	0.078
word2vec	0.103	0.075	0.075
fasttext-crawl-sub	0.092	0.070	0.069
bert-base	0.020	0.017	0.016
wordnet2vec	0.105	0.077	0.076
elmo	0.067	0.051	0.050

Table 3: Effect of predicting different numbers of randomly selected voxels.

embeddings	**nFix**	**TRT**	**FFD**
glove-300	0.010	0.017	0.027
word2vec	0.009	0.010	0.016
fasttext-crawl-sub	0.008	0.007	0.012
bert-base	0.005	0.003	0.004
wordnet2vec	0.010	0.010	0.019
elmo	0.008	0.009	0.011
average	**0.008**	**0.009**	**0.015**

Table 4: Comparison of word embeddings predicting single eye-tracking features: number of fixations (nFix), first fixation duration (FFD) and total reading time of a word (TRT).

5.2 Multiple hypotheses testing

With the purpose of achieving consistency and going towards a global quality metric that can be combined with other evaluation methods, we perform statistical significance testing on each hypothesis. A hypothesis consists of comparing the combination of an embedding type and a cognitive data source to the random baseline.

Since the distribution of our test data is unknown and the datasets are small, we perform a Wilcoxon signed-rank test for each hypothesis (Dror et al., 2018). Additionally, to counteract the multiple hypotheses problem, we apply the conservative Bonferroni correction, where the global null hypothesis is rejected if $p < \alpha/N$, where N is the number of hypotheses (Dror et al., 2017). In our setting, $\alpha = 0.01$ and $N = 4$ for EEG (one hypothesis per EEG data source), $N = 59$ for for fMRI (one hypothesis per participant of each fMRI data source), and $N = 42$ for eye-tracking (one hypothesis per feature per eye-tracking corpus).

This approach of significance testing can easily be used in combination with other intrinsic and extrinsic evaluation methods. The significance ratios

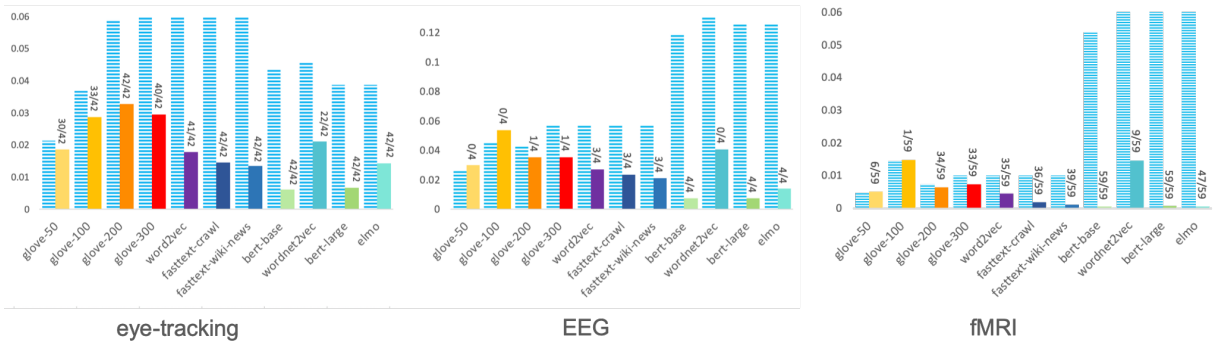


Figure 3: Results for each modality: Aggregated results for all embeddings predicting cognitive features for all datasets of a modality (sorted by dimension of embeddings in increasing order from left to right). The striped blued bars represent random baseline. The labels on the embedding bars show the ration of significant results under the Bonferroni correction to the total number of hypotheses.

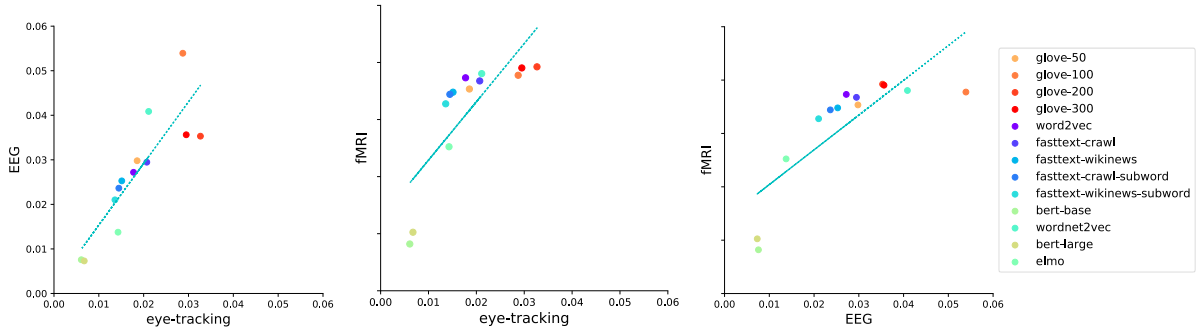


Figure 4: Correlation plots between all three modalities of cognitive signals.

are shown in Figure 3.

6 Results & Discussion

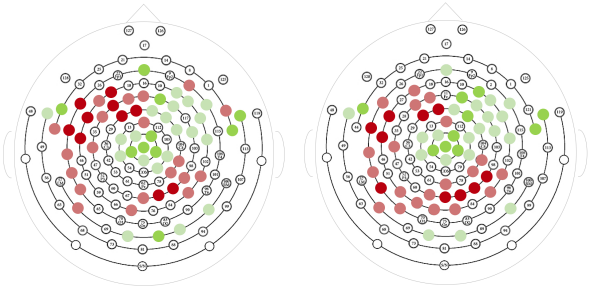
Prediction results First, we show in Figure 3 how well each word embedding type is able to predict eye-tracking features, EEG and fMRI vectors. As can be seen the majority of results are significantly better than the random baselines. BERT, ELMo and FastText embeddings achieve the best prediction results. All exact numbers can be found in Appendix B. While a random baseline can be considered a rather naive choice, this setting also allows us compare the performance between word embedding types.

When predicting single eye-tracking features, the performance varies greatly. For instance, Table 4 shows that the prediction error on number of fixations and total reading time from the ZuCo dataset is much lower than for first fixation duration. This suggest that more general eye-tracking features covering the complete reading process of a word are easier to predict.

In the case of predicting voxel vectors of fMRI data, the results improve when choosing a larger number of voxels (see Table 3). Hence, in the remainder of this work we present only the results for 1000 voxels.

We also examined the EEG results in more depth by analyzing which electrodes are predicted more accurately and which electrodes values are very difficult to predict. This is exemplified by Figure 5, which shows the 20 best and worst predicted electrodes of the ZuCo data for the BERT embeddings of 1024 dimensions as well as aggregated over all cognitive data sources. The middle central electrodes are predicted more accurately. The middle central electrodes are known to register the activity of the Perisylvian cortex, which is relevant for language related processing (Catani et al., 2005). Moreover it can be speculated that there is a frontal asymmetry between the electrodes on the left and right hemispheres.

Cognitive data implications The diversity of cognitive data sources chosen for this work allows us to analyze and compare results on several levels and between several cognitive metrics. In order to conduct this evaluation on a collection of 15



(a) (b)

Figure 5: EEG electrode analysis, (a) for BERT (large) and (b) aggregated over all embedding types. Red = worst predicted electrodes, green = best predicted electrodes.

datasets from three modalities, many crucial decisions were taken about preprocessing, feature extraction and evaluation type. Since there are different methods on how to process different types of cognitive language understanding signal, it is important to make these decisions transparent and reproducible.

Moreover, it is a challenge to segment brain activity data correctly and meaningfully into word-level signal from naturalistic, continuous language stimulus (Hamilton and Huth, 2018). This makes consistent preprocessing across data sources even more important.

Another challenge is to consolidate the cognitive features to be predicted. For instance, we chose a wide selection of eye-tracking features that cover early and late word processing. However, choosing only general eye-tracking features such as total reading time would also be a viable strategy. On the other hand, the EEG evaluation could be more coarse-grained, one could also try to predict known ERP effects (e.g. Ettinger et al. (2016)) or features selected based on frequency bands. Moreover, the voxel selection in the fMRI preprocessing could be improved by either predicting all voxels or applying information-driven voxel selection methods (Beinborn et al., 2019).

Correlations between modalities Next, we analyze the correlation between the predictions of the three modalities (Figure 4). There is a strong correlation between the results of predicting eye-tracking, EEG and fMRI features. This implies that word embeddings are actually predicting brain activity signals and not merely preprocessing artifacts of each modality. Moreover, the

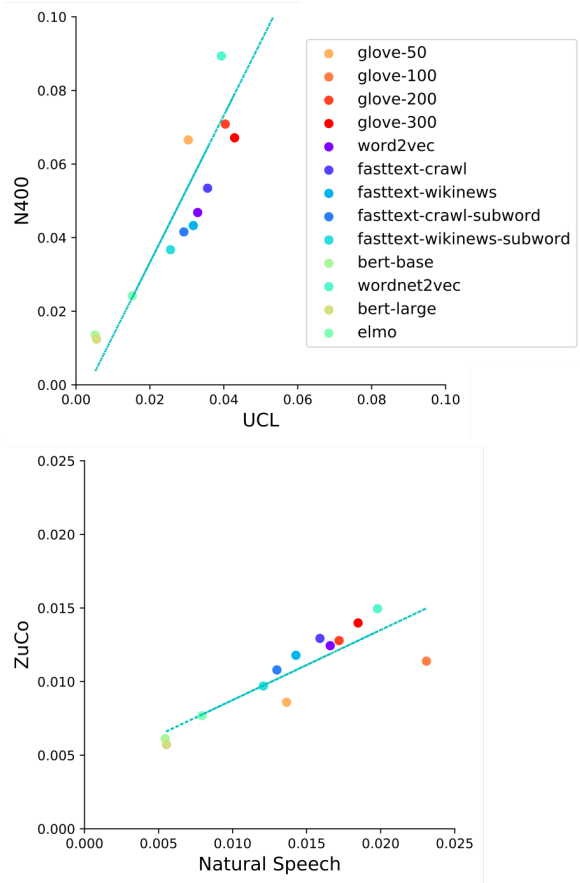


Figure 6: Correlation between results on EEG datasets.

same correlation is also evident between individual datasets within the same modality. As an example, Figure 6 (bottom) shows the correlation of the results predicted for the Natural Speech and ZuCo EEG datasets, where the first had speech stimuli and the latter text. Figure 6 (top) reveals the same positive correlation for two EEG datasets that were preprocessed differently and were recorded with a different number of electrodes. Moreover, the UCL dataset contains word-by-word reading and the N400 contains natural reading of full sentences.

Correlation with extrinsic evaluation results We performed a simple comparison between the results of word embeddings predicting cognitive language processing signals and the performance of the same embedding types in downstream tasks. We collected results for two NLP tasks: on the SQuAD 1.1 dataset for question answering (Rajpurkar et al., 2016) and on the CoNLL-2003 test split for named entity recognition (Tjong Kim Sang and De Meulder, 2003).

The SQuAD results are taken from Devlin et al.

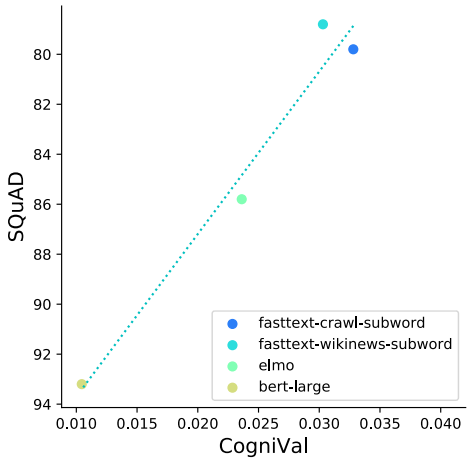


Figure 7: Correlation between the SQuAD 1.1 task and the CogniVal results.

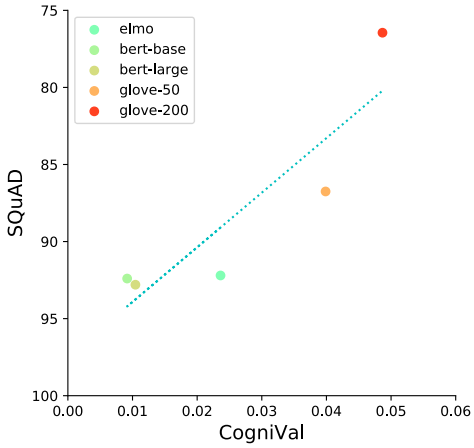


Figure 8: Correlation between NER on CoNLL-2003 and the CogniVal results.

(2019) for BERT, from Mikolov et al. (2018) for FastText, and from Peters et al. (2018) for ELMo. The NER results are from the same source for ELMo and BERT, for Glove-50 from Pennington et al. (2014) and for Glove-200 from Ghannay et al. (2016). We correlated these results to the prediction results over all cognitive data sources. Figures 7 and 8 show the correlation plots between the CogniVal results and the two downstream tasks.

While this is merely an exploratory analysis, it shows interesting findings: If the cognitive embedding evaluation correlates with the performance of the embeddings in extrinsic evaluation tasks, it might be used not only for evaluation but also as a predictive framework for word embedding model selection. This is especially noteworthy, since it does not seem to be the case for other intrinsic methods (Chiu et al., 2016).

7 Conclusion

We presented CogniVal, the first multi-modal large-scale cognitive word embedding evaluation framework. The vectorized word representations are evaluated by using them to predict eye-tracking or brain activity data recorded while participants were understanding natural language. We find that the results of eye-tracking, EEG and fMRI data are strongly correlated not only across these modalities but even between datasets within the same modality. Intriguingly, we also find a correlation between our cognitive evaluation and two extrinsic NLP tasks, which opens the question whether CogniVal can also be used for predicting downstream performance and hence, choosing the best embeddings for specific tasks.

We plan to expand the collection of cognitive data sources as more of them become available, including data from other languages such as the Narrative Brain Dataset (Dutch, fMRI, Lopopolo et al. (2018)) or the Russian Sentence Corpus (eye-tracking, Laurinavichyute et al. (2017)). Thanks to naturalistic recording of longer text spans, CogniVal can also be extended to evaluate sentence embeddings or even paragraph embeddings.

CogniVal can become even more effective by combining the results with other intrinsic or extrinsic embedding evaluation frameworks (Nayak et al., 2016; Rogers et al., 2018) and building on the multiple hypotheses testing.

8 Acknowledgements

We thank Lisa Beinborn, Stefan Frank and Thomas Lemmin for their valuable input on preprocessing EEG and fMRI data.

References

Samira Abnar, Rasyan Ahmed, Max Mijnheer, and Willem Zuidema. 2018. Experiential, distributional and dependency-based word embeddings have complementary roles in decoding brain activity. In *Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018)*, pages 57–66.

Amir Bakarov. 2018a. Can eye movement data be used as ground truth for word embeddings evaluation? *arXiv preprint arXiv:1804.08749*.

Amir Bakarov. 2018b. A survey of word embeddings evaluation methods. *arXiv preprint arXiv:1801.09536*.

Lisa Beinborn, Samira Abnar, and Rochelle Choenni. 2019. Robust evaluation of language-brain encoding experiments. *arXiv preprint arXiv:1904.02547*.

Jonathan R Brennan, Edward P Stabler, Sarah E Van Wagenen, Wen-Ming Luh, and John T Hale. 2016. Abstract linguistic structure correlates with temporal activity during naturalistic comprehension. *Brain and Language*, 157:81–94.

Michael P Broderick, Andrew J Anderson, Giovanni M Di Liberto, Michael J Crosse, and Edmund C Lalor. 2018. Electrophysiological correlates of semantic dissimilarity reflect the comprehension of natural, narrative speech. *Current Biology*, 28(5):803–809.

Marco Catani, Derek K Jones, and Dominic H Ffytche. 2005. Perisylvian language networks of the human brain. *Annals of Neurology: Official Journal of the American Neurological Association and the Child Neurology Society*, 57(1):8–16.

Billy Chiu, Anna Korhonen, and Sampo Pyysalo. 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 1–6.

Uschi Cop, Nicolas Dirix, Denis Drieghe, and Wouter Duyck. 2017. Presenting GECO: An eyetracking corpus of monolingual and bilingual sentence reading. *Behavior research methods*, 49(2):602–615.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Rotem Dror, Gili Baumer, Marina Bogomolov, and Roi Reichart. 2017. Replicability analysis for natural language processing: Testing significance with multiple datasets. *Transactions of the Association for Computational Linguistics*, 5:471–486.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392.

Allyson Ettinger, Naomi Feldman, Philip Resnik, and Colin Phillips. 2016. Modeling N400 amplitude using vector space models of word representation. In *CogSci*.

Stefan L Frank. 2017. Word embedding distance does not predict word reading time.

Stefan L Frank, Irene Fernandez Monsalve, Robin L Thompson, and Gabriella Vigliocco. 2013. Reading time data for evaluating broad-coverage models of english sentence processing. *Behavior Research Methods*, 45(4):1182–1190.

Stefan L Frank, Leun J Otten, Giulia Galli, and Gabriella Vigliocco. 2015. The ERP response to the amount of information conveyed by words in sentences. *Brain and language*, 140:1–11.

Stefan L Frank and Roel M Willems. 2017. Word predictability and semantic similarity show distinct patterns of brain activity during language comprehension. *Language, Cognition and Neuroscience*, 32(9):1192–1203.

Sahar Ghannay, Benoit Favre, Yannick Esteve, and Nathalie Camelin. 2016. Word embedding evaluation and combination. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 300–305.

Anna Gladkova and Aleksandr Drozd. 2016. Intrinsic evaluations of word embeddings: What can we do better? In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 36–42.

Liberty S Hamilton and Alexander G Huth. 2018. The revolution will not be controlled: Natural stimuli in speech neuroscience. *Language, Cognition and Neuroscience*, pages 1–10.

Olaf Hauk and Friedemann Pulvermüller. 2004. Effects of word length and frequency on the human event-related potential. *Clinical Neurophysiology*, 115(5):1090–1103.

Nora Hollenstein, Jonathan Rotsztejn, Marius Troendle, Andreas Pedroni, Ce Zhang, and Nicolas Langer. 2018. ZuCo, a simultaneous EEG and eyetracking resource for natural sentence reading. *Scientific Data*.

Nora Hollenstein and Ce Zhang. 2019. Entity recognition at first sight: Improving NER with eye movement information. In *NAACL*.

Alexander G Huth, Wendy A de Heer, Thomas L Griffiths, Frédéric E Theunissen, and Jack L Gallant. 2016. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453–458.

Marcel A Just and Patricia A Carpenter. 1980. A theory of reading: From eye fixations to comprehension. *Psychological review*, 87(4):329.

Alan Kennedy, Robin Hill, and Joël Pynte. 2003. The Dundee corpus. In *Proceedings of the 12th European Conference on Eye Movement*.

AK Laurinavichyute, Irina A Sekerina, SV Alexeeva, and KA Bagdasaryan. 2017. Russian Sentence Corpus: Benchmark measures of eye movements in reading in Cyrillic.

Alessandro Lopopolo, Stefan L Frank, Antal Van den Bosch, Annabel Nijhof, and Roel M Willems. 2018. The Narrative Brain Dataset (NBD), an fMRI dataset for the study of natural language processing in the brain. In *LREC 2018 Workshop on Linguistic and Neuro-Cognitive Resources (LiNCR)*. LREC.

Steven G Luke and Kiel Christianson. 2017. The Provo Corpus: A large eye-tracking corpus with predictability norms. *Behavior Research Methods*, pages 1–8.

Rob van der Goot Malvina Nissim, Rik van Noord. 2019. Fair is better than sensational: Man is to doctor as woman is to doctor. *arXiv preprint arXiv:1905.09866*.

Francis M Miezin, L Maccotta, JM Ollinger, SE Petersen, and RL Buckner. 2000. Characterizing the hemodynamic response: effects of presentation rate, sampling procedure, and the possibility of ordering brain activity based on relative timing. *Neuroimage*, 11(6):735–759.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller and Christiane Fellbaum. 1992. Wordnet and the organization of lexical memory. In *Intelligent tutoring systems for foreign language learning*, pages 89–102. Springer.

Abhijit Mishra, Diptesh Kanojia, and Pushpak Bhattacharyya. 2016. Predicting readers' sarcasm understandability by modeling gaze behavior. In *AAAI*, pages 3747–3753.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2017. Scanpath complexity: Modeling reading effort using gaze information. In *AAAI*, pages 4429–4436.

Tom M Mitchell, Svetlana V Shinkareva, Andrew Carlson, Kai-Min Chang, Vicente L Malave, Robert A Mason, and Marcel Adam Just. 2008. Predicting human brain activity associated with the meanings of nouns. *Science*, 320(5880):1191–1195.

Christoph Mulert. 2013. Simultaneous EEG and fMRI: towards the characterization of structure and dynamics of brain networks. *Dialogues in clinical neuroscience*, 15(3):381.

Brian Murphy, Leila Wehbe, and Alona Fyshe. 2018. Decoding language from the brain. *Language, cognition, and computational models*, page 53.

Neha Nayak, Gabor Angeli, and Christopher D Manning. 2016. Evaluating word embeddings using a representative suite of practical tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 19–23.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Francisco Pereira, Bin Lou, Brianna Pritchett, Samuel Ritter, Samuel J Gershman, Nancy Kanwisher, Matthew Botvinick, and Evelina Fedorenko. 2018. Toward a universal decoder of linguistic meaning from brain activation. *Nature communications*, 9(1):963.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*.

Cathy J Price. 2012. A review and synthesis of the first 20 years of PET and fMRI studies of heard speech, spoken language and reading. *Neuroimage*, 62(2):816–847.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Keith Rayner. 1998. Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, 124(3):372.

Joao António Rodrigues, Ruben Branco, João Silva, Chakaveh Saedi, and António Branco. 2018. Predicting brain activation with WordNet embeddings. In *Proceedings of the Eight Workshop on Cognitive Aspects of Computational Language Learning and Processing*, pages 1–5.

Anna Rogers, Shashwath Hosur Ananthakrishna, and Anna Rumshisky. 2018. What's in your embedding, and how it predicts task performance. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2690–2703.

Chakaveh Saedi, António Branco, João António Rodrigues, and João Silva. 2018. WordNet embeddings. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 122–131.

Dan Schwartz and Tom Mitchell. 2019. Understanding language-elicited EEG data by predicting it from a fine-tuned language model. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 43–57.

Anders Søgaard. 2016. Evaluating word embeddings with fMRI and eye-tracking. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP*, pages 116–121.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*, volume 4, pages 142–147.

Leila Wehbe, Ashish Vaswani, Kevin Knight, and Tom Mitchell. 2014. Aligning context-based statistical models of language with brain activity during reading. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 233–243.

KnowSemLM : A Knowledge Infused Semantic Language Model

Haoruo Peng[1], **Qiang Ning**[1], **Dan Roth**[1,2]
Department of Computer Science
[1]University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
[2]University of Pennsylvania, Philadelphia, PA 19104, USA
{hpeng7,qning2}@illinois.edu, danroth@seas.upenn.edu

Abstract

Story understanding requires developing expectations of what events come next in text. Prior knowledge – both statistical and declarative – is essential in guiding such expectations. While existing semantic language models (SemLM) capture event co-occurrence information by modeling event sequences as semantic frames, entities, and other semantic units, this paper aims at augmenting them with causal knowledge (i.e., one event is likely to lead to another). Such knowledge is modeled at the frame and entity level, and can be obtained either statistically from text or stated declaratively. The proposed method, KnowSemLM[1], infuses this knowledge into a semantic LM by joint training and inference, and is shown to be effective on both the event cloze test and story/referent prediction tasks.

1 Introduction

Natural language understanding requires a coherent understanding of a series of events or actions in a story. In story comprehension, we need to understand not only what events have appeared in text, but also what is likely to happen *next*. While event extraction has been well studied (Ji and Grishman, 2008; Huang and Riloff, 2012; Li et al., 2013; Peng et al., 2016; Nguyen et al., 2016; Nguyen and Grishman, 2016), the task of predicting future events (Radinsky et al., 2012; Radinsky and Horvitz, 2013) has received less attention.

One perspective is to utilize the co-occurrence information between past and future events learned from a large corpus, which has been studied in *script learning* works (Chambers and Jurafsky, 2008; Pichotta and Mooney, 2014, 2016a; Peng and Roth, 2016; Peng et al., 2017). However, only considering co-occurrence information is not sufficient for modeling event sequences in natural language. Human decisions on the likelihood of a specific event depend on both *local context* – what has happened earlier in text – and *global context* – knowledge gained from human experience. This paper leverages both the *local* and *global* context information to model event sequences, and shows that it can lead to more accurate predictions of future events. For example, the following text snippet describes a scenario of someone taking a flight:

... I checked in at the counter, took my luggage to the security area, got cleared ten minutes in advance, and waited for my plane ...

This example consists of a series of events, i.e., "check in (a flight)", "be cleared (at the security)", "wait for (the plane)", etc., which humans who have traveled by plane are very familiar with. However, this event sequence appears infrequently in text.[2] Consequently, only relying on event co-occurrence in text is not sufficient – there is also a need to model some "common sense" information.

The local and global contexts in this example are illustrated in Figure 1. The existing event sequence is "(sub)check_in[flight]", "(sub)clear[security]" and "(sub)wait_for[plane]" (denoted by blue dots), where "sub" means subject. Language models (LM) for statistical co-occurrences of events can capture this local context and generate a distribution over all possible events, e.g., "(sub)purchase[food]" and "(sub)go_to[work]", as in the blue circle.

More importantly, global context is the knowledge of event causality learned from human experience in the form of "cause-effect" event pairs (i.e., one event leads to another). One such pair is represented as "(sub)wait_for[plane] ⇒

[1]Related resources refer to https://cogcomp.seas.upenn.edu/page/publication_view/886.

[2]The events "check in" and "be cleared" only co-occur twice in a same document in the 20-year New York Times corpus (1987-2007); we count with frame and entity level abstractions (see Section 2.1 for details).

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 550–562
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

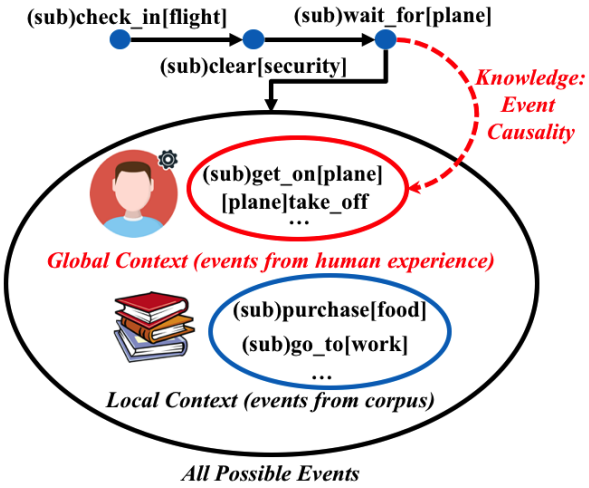


Figure 1: **Local and global context information when modeling event sequences.** The blue dots are events that are already described in text. The blue circle indicates local context, i.e., event sequences inferred from a large corpus via semantic LMs; the red circle represents global context, i.e., events learned from human experience via knowledge of event causality (which may overlap with local context). For event representations, we abstract over the surface forms of semantic frames and entities, where "sub" represents the shared common subject. The proposed KnowSemLM leverages both information to better predict future events.

(sub)get_on[plane]", which means that one has to wait for a plane before getting on it (red dashed arrow in Figure 1). Global context, as a result, helps generate a distribution over a focused set of expected events, as in the red circle. Note that the causality links have directions, and one event might lead to multiple possible events, e.g., one has to wait for the plane before it takes off "(sub)wait_for[plane] ⇒ [plane]take_off". Such connections can be viewed as temporal relations. Here, we consider causality to include temporal orderings of events which align with common sense. More discussions are provided in Sec. 6.

Thus, we propose KnowSemLM, a knowledge infused semantic language model. It combines knowledge from external sources (in the form of event causality) with the basic semantic LM (Peng et al., 2017) trained on a given text corpus. Our model is a generative model of events, where each event is either generated based on a piece of knowledge or generated from the semantic LM. When predicting future events at inference time, we generate two distributions over events: one from the given knowledge, and the other from the semantic LM. We also learn a binary variable that selects the distribution from which we take the next event. In this way, the proposed KnowSemLM has the ability to generate event sequences based on both local and global context, and better imitate the story generation process.

This knowledge infused semantic LM operates on abstractions over the surface form – semantic frames and entities. We associate each semantic unit (frames and entities) with an embedding and construct a joint embedding space for each event. We train KnowSemLM on a large corpus and use the same embedding setting for events involved in the knowledge. The event causality knowledge is mined either statistically from the training corpus or declaratively for constrained domains (both in the form of event pairs). In the statistical way, we utilize a set of discourse connectives to identify "cause-effect" event pairs and filter them based on their counts; if provided with event templates for specific domains, we also manually write down such pairs based on human experience. In both ways, we further enrich the knowledge base by considering transitivity among event pairs.

We evaluate KnowSemLM on two tasks – event cloze test and story/referent predictions. In both cases, we model text as a sequence of events and apply trained KnowSemLM to calculate conditional probabilities of future events given text and knowledge. We show that KnowSemLM can outperform competitive results from models with no such knowledge. In addition, we demonstrate the language modeling ability of KnowSemLM through quantitative and qualitative analysis.

The main contributions can be summarized as follows: 1) formulation of knowledge used in story generation as event causality; 2) proposal of KnowSemLM to integrate such event causality knowledge into semantic language models; 3) demonstration of the effectiveness of KnowSemLM via multiple benchmark tests.

The rest of the paper is organized as follows. We define how we model events and event causality knowledge in Sec. 2, followed by the description of the knowledge infused KnowSemLM (Sec. 3). The training procedure of KnowSemLM is detailed in Sec. 4, followed by our experimental results and analysis (Sec. 5) and related work (Sec. 6). We conclude in Sec. 7.

2 Event and Knowledge Modeling

To better understand the proposed KnowSemLM, here we first introduce the event representation and

event causality model used in this paper.

2.1 Event Representation

To preserve the full semantic meaning of events, we need to consider multiple semantic aspects: semantic frames, entities, and sentiments. We adopt the event representation proposed in Peng et al. (2017), which is built upon abstractions of three basic semantic units: (disambiguated) semantic frames, subjects & objects in such semantic frames, and sentiments of the frame text.

In a nutshell, the event representation is a combination of the above three semantic elements.

... Steven Avery committed murder. He was arrested, charged and tried ...

For example, the event representations of the above text would be (four separate events):

PER[new]-commit.01-ARG[new](NEG)
ARG[new]-arrest.01-PER[old](NEU)
ARG[new]-charge.05-PER[old](NEU)
ARG[new]-try.01-PER[old](NEG)

Here, "commit.01", "arrest.01" and so on represent disambiguated predicates ("01" and "05" refer to the disambiguated senses in VerbNet). The arguments (subject and object) of a predicate are denoted with NER types ("PER, LOC, ORG, MISC") or "ARG" if unknown, along with a "[new/old]" label indicating if it is the first appearance in the sequence. Additionally, the sentiment of a frame is represented as positive (POS), neural (NEU), or negative (NEG).

We formally define such an explicit and abstracted event as e. Computationally, the vector representation of an event e^{vec} is built in a joint semantic space:

$$e^{\text{vec}} = W_f r_f + W_e r_e + W_s r_s.$$

During language model training, we learn frame embeddings W_f (r_f, r_e, r_s are one-hot vectors for each unique frame, entity and sentiment abstraction, respectively) as well as the transforming matrices W_e and W_s.

2.2 Knowledge: Causality between Events

We model the knowledge gained from human experience as pre-determined relationship between events. Since we are modeling event sequences, the knowledge of one event leads to another is very important, hence event causality. We formally define a piece of event knowledge as

$$e_x \Rightarrow e_y,$$

meaning that the *outcome* event e_y is a possible result of the *causal* event e_x. Note that event causality here is directional, and one event may lead to multiple different outcomes. We group all event knowledge pairs with the same causal event, thus event e_x can lead to a set of events:

$$e_x \Rightarrow \{e_{y_1}, e_{y_2}, e_{y_3}, \cdots, e_{y_m}\}.$$

We store all such event causality structures in a knowledge base KB_{EC}.

3 Knowledge Infused SemLM

With a proper modeling of events and event causality above, this section explains the proposed `KnowSemLM`, a method to inject causality knowledge into a semantic LM. Specifically, `KnowSemLM` is based on FES-RNNLM (*Frame-Entity-Sentiment infused Recurrent Neural Net Language Model*) proposed in Peng et al. (2017). We briefly review FES-RNNLM and describe how `KnowSemLM` adds knowledge on top of it.

3.1 FES-RNNLM

To model semantic sequences and train the joint event representations in Sec. 2.1, we build neural language models over such sequences. Peng et al. (2017) uses Log-Bilinear Language model (Mnih and Hinton, 2007), but since we require the use of event causality knowledge to be based on past events, we choose to implement an RNN language model (RNNLM) where the generation of future events is only dependent on past events.

For ease of explanation, we denote a semantic sequence of joint event representations as $[e_1, e_2, \cdots, e_t]$, with e_t being the t_{th} event in the sequence. Thus, we model the conditional probability of an event e_t given its context as

$$\begin{aligned} & p_{\text{lm}}(e_t | e_1, \cdots, e_{t-1}) \\ &= \text{softmax}(W_s h_t + b_s) \\ &= \frac{\exp(e_t^{\text{vec}}(W_s h_t + b_s))}{\sum_{e \in \mathcal{V}} \exp(e^{\text{vec}}(W_s h_t + b_s))}. \end{aligned}$$

Note that the softmax operation is carried out over the event vocabulary $\mathcal{V}$, i.e., all possible events in the language model. Moreover, the hidden layer h_t in RNN is computed as: $h_t = \phi(e_t^{\text{vec}} W_i + h_{t-1} W_h + b_h)$, where ϕ is the activation function. For language model training, we learn parameters W_s, b_s, W_i, W_h, and b_h, and maximize the sequence probability $\prod_{t=1}^{k} p_{\text{lm}}(e_t | e_1, e_2, \cdots, e_{t-1})$.

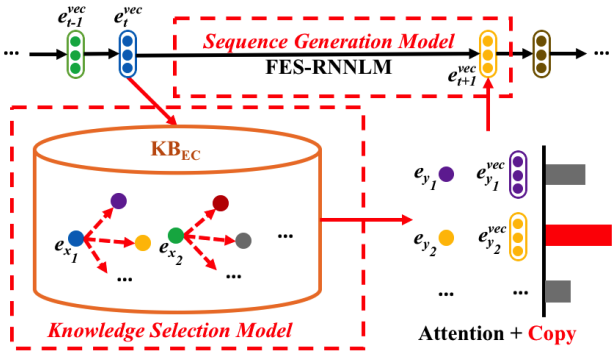


Figure 2: **Overview of the computational workflow for the proposed KnowSemLM.** There are two key components: 1) a knowledge selection model, which activates the use of knowledge based on probabilistically matching causal event and produce a distribution over outcome events via attention; 2) a sequence generation model, which takes input from both the knowledge selection model and the base semantic language model (FES-RNNLM) to generate future events via a copying mechanism. Note that the single dots indicate explicit event representations while three consecutive dots stand for event vectors.

3.2 KnowSemLM

In Figure 2, we show the computational workflow of the proposed KnowSemLM. There are two key components: 1) a knowledge selection model, which activates the use of knowledge based on probabilistically matching causal events and produces a distribution over outcome events; 2) a sequence generation model, which takes input from both the knowledge selection model and the base semantic language model (FES-RNNLM) to generate future events via a copying mechanism.[3]

Knowledge Selection Model

For an event in the sequence e_t, we first match it with possible causal events $\{e_x\}$ in the knowledge base KB_{EC} based on the bi-focal attention of previous events. Thus, from the knowledge base, we get a list of outcome events $\mathcal{V}_y \triangleq \{e_{y_1}, e_{y_2}, \cdots\}$.

Computationally, we model the conditional probability of matching with causal event e_x and outcome event e_y from knowledge base given the context of $e_1, e_2, \cdots, e_t$ as

$$p_{\text{kn}}(e_x \Rightarrow e_y | e_1, e_2, \cdots, e_t) = \frac{\exp(e_x^{\text{vec}} W_a h_t) \exp(e_y^{\text{vec}} W_b h_t)}{\sum_{e \in \mathcal{V}_x, e' \in \mathcal{V}_y} \exp(e^{\text{vec}} W_a h_t) \exp(e'^{\text{vec}} W_b h_t)}.$$

[3]The proposed computational framework of KnowSemLM is similar to DynoNet proposed in He et al. (2017). Compared to DynoNet, the knowledge base utilized here operates on event level representations rather than on tokens.

Here, we use the bi-focal attention mechanism (Nema et al., 2018) via attention parameters W_a, W_b, and apply it on the hidden layer h_t, which embeds information from all previous events in the sequence. Therefore, we produce a distribution over the set of possible outcome events $\mathcal{V}_y$.

Sequence Generation Model

The base semantic LM produces a distribution over events from the language model vocabulary, which represents *local* context, while the knowledge selection model generates a set of outcome events with a probability distribution, which represents *global* context of event causality knowledge. The sequence generation model then combines the local and global context for generating future events. Therefore, we model the conditional probability of event e_{t+1} given context $p(e_{t+1}|\text{Context}) = p(e_{t+1}|e_1, e_2, \cdots, e_t, \text{KB}_{\text{EC}})$. This overall distribution is computed via a copying mechanism (Jia and Liang, 2016), i.e., we either generate the next event (e_i) from the language model vocabulary ($\mathcal{V}$) or copy from the outcome event set (e_y) based on the following probabilities:

$$\begin{cases} p(e_{t+1} = e_i \in \mathcal{V} | \text{Context}) & = (1-\lambda) p_{\text{lm}}(e_i) \\ p(e_{t+1} = e_y \in \mathcal{V}_y | \text{Context}) & = \lambda p_{\text{kn}}(e_y). \end{cases}$$

Here, λ is a learned scaling parameter to choose between events from LM vocabulary $\mathcal{V}$ and events from event causality knowledge base KB_{EC}.

4 Construction of KnowSemLM

4.1 Dataset and Preprocessing

Dataset: We use the New York Times (NYT) Corpus[4] (from year 1987 to 2007) as the training corpus. It contains over 1.8M documents in total.

Preprocessing: We preprocess all training documents with Semantic Role Labeling and Part-of-Speech tagging. We also implement the explicit discourse connective identification module of a shallow discourse parser (Song et al., 2015). Additionally, we utilize within-document entity co-reference (Peng et al., 2015a) to produce co-reference chains and get the anaphoricity information. To obtain all annotations, we use the Illinois NLP tools (Khashabi et al., 2018).[5] Further, we obtain event representations from text with frame, entity and sentiment level abstractions by following procedures described in Peng et al. (2017).

[4]https://catalog.ldc.upenn.edu/LDC2008T19

[5]http://cogcomp.org/page/software/

4.2 Knowledge Mining

Statistical Way: Part of the human knowledge can be mined from text itself. Since discourse connectives are important for relating different text spans, we carefully select discourse connectives which can indicate a "cause-effect" situation. For example, "The police arrested Jack *because* he killed someone." In this sentence, readers can gain the knowledge of "the person who kills shall be arrested", which can be represented as "PER[*]-kill.01-*[*](*) ⇒ *[*]-arrest.01-PER[old](*)" according to the abstractions specified in Sec. 2.

In practice, we choose 22 "cause-effect" connectives/phrases (such as "because", "due to", "in order to"). We then extract all event pairs connected by such connectives from the NYT training data, and abstract over their surface forms to get the event level representations. Finally, we filter cases where the direction of the event causality pairs is unclear from a statistical standpoint. Specifically, we calculate the ratio of counts of one direction over another, i.e. $\theta = \frac{\#(e_x \Rightarrow e_y)}{\#(e_y \Rightarrow e_x)}$. If $\theta > 2$, then we store $e_x \Rightarrow e_y$ as knowledge; while $\theta < 0.5$, we only keep $e_y \Rightarrow e_x$. In the case of $0.5 < \theta < 2$, we filter both event causality pairs since we are unsure of the knowledge statistically.

After the above filtering procedures, we automatically get 8,293 different pairs of event pairs (without human efforts). According to Sec. 2, we merge them if they have the same causal event, i.e. $e_x \Rightarrow e_y$ and $e_x \Rightarrow e_z$ becomes $e_x \Rightarrow \{e_y, e_x\}$. Thus, we get a total of 2,037 causal events (trees); and on average, each causal event has 4 possible outcome events. Furthermore, those event pairs of knowledge defined in this work are transitive, e.g., if $e_1 \Rightarrow e_2$ and $e_2 \Rightarrow e_3$, then we can have $e_1 \Rightarrow e_3$. Considering this *transitivity*, we iterate over all pairs twice, and derive more event causality pairs, achieving a total number of 9,022.[6]

Declarative Way: Besides mining knowledge automatically from text corpus, we also take full advantage of human input in some practical situations. For the InScript Corpus (Modi et al., 2017), it specifies 10 everyday scenarios, e.g., "Bath", "Flight", "Haircut". In each scenario, the corpus also provides event templates and the corresponding event template annotations for the text. Examples of such generated event causality knowledge can be referred back to Sec. 1, e.g., "(sub)wait_for[plane] ⇒ (sub)get_on[plane]". In total, we manually generate 875 event causality pairs and group them with 121 causal events. Here, since during the manual generation process, we try to cover all event causality knowledge that makes sense; we do not further apply the transitive property and expand.

[6]We do not further carry out the transitivity expansion process, since empirically the noise it introduces outweighs the benefits it brings (see Sec. 5.4 for details).

Method	Accuracy
Granroth-Wilding and Clark (2016)	49.57%
Wang et al. (2017)	55.12%
KnowSemLM w/o knowledge	39.23%
KnowSemLM w/o transit. & fine-tuning	43.56%
KnowSemLM w/o fine-tuning	45.28%
KnowSemLM	**56.27%**

Table 1: **Accuracy results for the event cloze task.** KnowSemLM outperforms previously reported results and we show the ablation study results for model without the use of knowledge (w/o knowledge), without the use of knowledge transitivity as described in Sec 4.2 (w/o transit.) and without fine-tuning on the dev data (w/o fine-tuning), resepctively.

4.3 Model Training

Based on the formulation in Sec. 3, we apply the overall sequence probability as the objective: $\prod_{t=1}^{k} p(e_t|e_1, e_2, \cdots, e_{t-1}, \text{KB}_{\text{EC}})$. where k is the sequence length. For the sequence generation model, we implement the Long Short-Term Memory (LSTM) network with a layer of 64 hidden units while the dimension of the input event vector representation is 200. Because we carry out the same event-level abstractions as in Peng et al. (2017), the event vocabulary is the same, with the size of ∼4M different events.[7]

5 Experiments

We show that KnowSemLM can achieve better performance for the event cloze test and story/referent prediction tasks compared to models without the use of knowledge. We also evaluate the language modeling ability of KnowSemLM through quantitative and qualitative analysis.

5.1 Application for Event Cloze Test

Task Description and Setting: We utilize the MCNC task and dataset proposed in Granroth-Wilding and Clark (2016) as the benchmark evaluation. For each test instance, the goal is to recover the event (defined as predicate with associated entities) from an event chain given multiple choices.

[7]Please see Table 2 in Peng et al. (2017) for details.

Since the event definition in this task is compatible with our representation defined in Sec 2.1[8], we can directly convert event chains into our semantic event sequences. In this application task, we train KnowSemLM on the NYT portion of the Gigaword[9] corpus, and also fine-tune on the development set specified in this task[10].

Application of KnowSemLM: For each test case (i.e., an event chain inside a document), we first construct the event level representation as described in Sec. 2 for each event in the chain. We then apply KnowSemLM to obtain the overall sequence probability by replacing the missing event with each candidate choice. The final decision is made by choosing the event with the highest probability. Note that the event causality knowledge here for both training and testing is generated automatically from NYT corpus specified in Sec. 4.2 (the Statistical Way). To efficiently calculate the sequence probability, we limit the context window size surrounding the missing event to be 10.

Results: The accuracy results are shown in Table 1. We compare KnowSemLM with previous reported results on this event cloze test (Granroth-Wilding and Clark, 2016; Wang et al., 2017). KnowSemLM outperforms both baselines and we further carry out the ablation study to measure the impact of knowledge, transitivity of knowledge, and fine-tuning. We can see that it is important for the semantic LM to consider knowledge and also learn the process of applying such knowledge in event sequences, i.e., the fine-tuning step.

5.2 Application for Story Prediction

Task Description and Setting: We use the benchmark ROCStories dataset (Mostafazadeh et al., 2017), and follow the test setting in Peng et al. (2017). For each instance, we are given a four-sentence story and the system needs to predict the correct fifth sentence from two choices; with the incorrect ending being semantically unreasonable, or un-related. Instead of treating the task as a supervised binary classification problem with a development set to tune, we evaluate KnowSemLM in an unsupervised fashion where we directly evaluate on the test set. In such a way, we can directly compare with the FES-LM model proposed in Peng et al. (2017), which is base model of KnowSemLM without the use of knowledge. Similar to the training of FES-LM, we fine tune KnowSemLM on the in-domain short story training data, with the model trained on NYT corpus as initialization.[11]

Baselines	Accuracy	
Seq2Seq	58.0%	
Mostafazadeh et al. (2016)	58.5%	
Seq2Seq with attention	59.1%	
Model w/o Knowledge	S.	M.V.
FES-LM (Peng et al., 2017)	62.3%	61.6%
Knowledge Model	S.	M.V.
KnowSemLM	**66.5%**	63.1%

Table 2: **Accuracy results for story cloze test in the unsupervised setting.** "S." represents the inference method with the single most informative feature while "M.V." means majority voting.

Application of KnowSemLM: For each test story, we generate a set of conditional probability features from KnowSemLM. We first construct the event level representation as described in Sec. 2. We then utilize the conditional probability of the fifth sentence given previous context sentences and the knowledge base KB_{EC} as features. Here KB_{EC} is generated automatically from NYT corpus specified in Sec. 4.2 without human efforts. We get multiple features depending on how long we go back in the context in terms of events. In practice, we get at most 12 events as context since one sentence can contain multiple events. Thus, for each story, we generate at most 12 pairs of conditional probability features from two given choices. Every pair of such features can yield a decision on which ending is more probable. Here, we test two different inference methods: a single most informative feature (where we go with the decision made by the pair of features which have the highest ratio) or majority voting based on the decision made jointly by all feature pairs.

Results: The accuracy results are shown in Table 2. We compare KnowSemLM with Seq2Seq baselines (Sutskever et al., 2014) and Seq2Seq with attention mechanism (Bahdanau et al., 2014). We also include the DSSM system

[8] Our event representation is abstracted on a higher level. Thus, we process the original NYT documents, where event chains come from, for abstraction purposes; and then match it to the event chains in the test data.

[9] https://catalog.ldc.upenn.edu/LDC2011T07

[10] https://mark.granroth-wilding.co.uk/papers/what_happens_next/

[11] We iterate over the NYT corpus until it converges on the perplexity metric for the development set, and then the model is further trained on ROC-Stories training set for 5 epochs.

Method	Accuracy
Base (Modi et al., 2017)	62.65%
EntityNLM (Ji et al., 2017)	74.23%
Base*	60.58%
Base* w/ FES-RNNLM	63.79%
Base* w/ KnowSemLM	**76.15%**

Table 3: **Accuracy results for the referent prediction task on InScript Corpus.** We re-implement the base model (Modi et al., 2017) as "Base*", and apply KnowSemLM to add additional features. "Base* w/ FES-RNNLM" is the ablation study where no event causality knowledge is used. Even though "Base*" model performs not as good as the original base model, we achieve the best performance with added KnowSemLM features.

Perplexity	
FES-RNNLM	121.8
KnowSemLM w/o transitivity	120.7
KnowSemLM	120.4
Narrative Cloze Test (Recall@30)	
FES-RNNLM	47.9
KnowSemLM w/o transitivity	49.3
KnowSemLM	49.6

Table 4: **Results for perplexity and narrative cloze test.** Both studies are conducted on the NYT hold-out data. "FES-RNNLM" represents the semantic LM without the use of knowledge. The numbers show that KnowSemLM has lower perplexity and higher recall on narrative cloze test, which demonstrates the contribution of the infused knowledge.

	Match/Event	Activation/Event	λ
NYT	0.13	0.03	0.36
InScript	0.82	0.28	0.46

Table 5: **Statistics for the use of event causality knowledge in KnowSemLM.** We gather the statistics for both NYT and InScript Corpus. "Match/Event" represents average number of times a causal event match is found in the event causality knowledge base per event; while "Activation/Event" stands for the average number of times we actually generate event predictions from the outcome events of the knowledge base. In addition, we believe the ratio of "Activation/Event" over "Match/Event" co-relates with the scaling parameter λ.

from Mostafazadeh et al. (2016) as the original reported result. KnowSemLM outperforms both baselines and the base model without the use of knowledge, i.e., FES-LM. The best performance achieved by KnowSemLM uses single most informative feature, with the feature being the conditional probability depending on only the nearest preceding event and event causality knowledge).

5.3 Application for Referent Prediction

Task Description and Setting: For referent prediction task, we follow the setting in Modi et al. (2017), where the system predicts the referent of an entity (or a new entity) given the preceding text. The task is evaluated on the InScript Corpus, which contains a group of documents where events are manually annotated according to predefined event templates. Each document contains one entity which needs to be resolved. The InScript Corpus can be divided into 10 situations and is split into standard training, development, and testing sets. We fine-tune KnowSemLM on the InScript Corpus training set, with the model trained on NYT corpus as initialization.

Application of KnowSemLM: For each test case (i.e., an entity inside a document), each candidate choice will be represented as a different event representation. Note that the event representation here comes from the event templates defined in the InScript Corpus. In the meantime, we can extract the event sequence from the preceding context. Thus, we can apply KnowSemLM to compute the conditional probability of the candidate event e_{t+1} given the event sequence and the event causality knowledge: $p_k(e_{t+1}|e_{t-k}, e_{t-k+1}, \cdots, e_t, \text{KB}_{\text{EC}})$. Here, knowledge in KB_{EC} is generated manually from event templates specified in Sec. 4.2. Moreover, index k decides how far back we consider the preceding event sequence. We then add this set of conditional probabilities as additional features in a base model (re-implementation of the linear model proposed in Modi et al. (2017), namely "Base*") to train a classifier to predict the right referent.

Results: The accuracy results are shown in Table 3. We compare with the original base model as well as the EntityNLM proposed in Ji et al. (2017) as baselines. Our re-implemented base model ("Re-base") does not perform as good as the original model. However, with the help of additional features from FES-RNNLM, we outperform the base model. More importantly, with additional features from KnowSemLM, we achieve the best performance and beat the EntityNLM system. This demonstrates the importance of the manually added event causality knowledge, and the ability of KnowSemLM to successfully capture it.

5.4 Analysis of KnowSemLM

First, to evaluate the language modeling ability of KnowSemLM, we report perplexity and narrative cloze test results. We employ the same experimental setting as detailed in Peng and Roth (2016) on the NYT hold-out data. Results are shown in Table 4. Here, "FES-RNNLM" serves as the semantic LM without the use of knowledge for the ablation study. The numbers shows that KnowSemLM has lower perplexity and higher recall on narrative cloze test; which demonstrates the contribution of the infused event causality knowledge. The results w.r.t. the transitivity evaluation shows that the expansion through knowledge transitivity improves the model quality.

We also gather the statistics to analyze the usage of event causality knowledge in KnowSemLM. We compute two key values: 1) average number of times a causal event match is found in the event causality knowledge base per event (so that we can potentially use the outcome events to predict), i.e. "Match/Event"; 2) average number of times we actually generate event predictions from the outcome events of the knowledge base (result of the final probability distribution), i.e. "Activation/Event". We get the statistics on both NYT and InScript Corpus, and associate the numbers with the scaling parameter λ in Table 5. The frequency of event matches and event activations from knowledge are both much lower in NYT than in InScript. Moreover, we can compute the chance of an outcome event being used as the prediction when it participates in the probability distribution. On NYT, it is $0.03/0.13 = 23\%$; while on InScript, it is $0.28/0.82 = 34\%$. We believe such chance co-relates with the scaling parameter λ.

For qualitative analysis, we provide a comparative example between KnowSemLM and FES-RNNLM in practice. The system is fed into the following input:

... Jane wanted to buy a new car. She had to borrow some money from her father. ...

So, on an event level, we abstract the text as "PER[new]-want.01-buy.01-ARG[new](NEU), PER[old]-have.04-borrow.01-ARG[new](NEU)". For FES-RNNLM, the system predicts the next event as "PER[old]-sell.01-ARG[new](NEU)" since in training data, there are many co-occurrences between the "borrow" event and "sell" event (coming from financial news articles in NYT). In contrast, for KnowSemLM, since we have the knowledge "PER[*]-borrow.01-ARG[*](*) $\Rightarrow$ PER[old]-return.01-ARG[old](*)", meaning that something borrowed by someone is likely to be returned, the predicted event would be "PER[old]-return.01-ARG[old](NEU)". This is closer to the real text semantically: ... *She promised to return the money once she got a job* ... Such an example shows that KnowSemLM works in situations where 1) the required knowledge is stored in the event causality knowledge base, and 2) the training data contains scenarios where required knowledge is put into use.

6 Related Work

Our work is built upon the previous works for semantic language models (Peng and Roth, 2016; Peng et al., 2017; Chaturvedi et al., 2017). This line of work is in general inspired by script learning. Early works (Schank and Abelson, 1977; Mooney and DeJong, 1985) tried to learn scripts via construction of knowledge bases from text. More recently, researchers focused on utilizing statistical models to extract high-quality scripts from large amounts of data (Chambers and Jurafsky, 2008; Bejan, 2008; Jans et al., 2012; Pichotta and Mooney, 2014; Granroth-Wilding and Clark, 2016; Rudinger et al., 2015; Pichotta and Mooney, 2016a,b). Other works aimed at learning a collection of structured events (Chambers, 2013; Cheung et al., 2013; Balasubramanian et al., 2013; Bamman and Smith, 2014; Nguyen et al., 2015; Inoue et al., 2016). In particular, Ferraro and Durme (2016) presented a unified probabilistic model of syntactic and semantic frames while also demonstrating improved coherence. Several works have employed neural embeddings (Modi and Titov, 2014a,b; Frermann et al., 2014; Titov and Khoddam, 2015). Some prior works have used scripts-related ideas to help improve NLP tasks (Irwin et al., 2011; Rahman and Ng, 2011; Peng et al., 2015b).

Several recent works focus on narrative/story telling (Rishes et al., 2013), as well as studying event structures (Brown et al., 2017). Most recently, Mostafazadeh et al. (2016, 2017) proposed story cloze test as a standard way to test a system's ability to model semantics. They released ROCStories dataset, and organized a shared task for LSDSem'17; which yields many interesting works on this task. Cai et al. (2017) developed a model that uses hierarchical recurrent networks with at-

tention to encode sentences and produced a strong baseline. Lee and Goldwasser (2019) considered the problem of learning relation aware event embeddings for commonsense inference, which can account for different relations between events, beyond simple event similarity. We differ from them because the basic semantic unit we model is event level abstractions instead of word tokens.

The definition of event causality knowledge in this work includes temporal ordering relationships. Much progress has been made in identifying and modeling such relations. In early works (Mani et al., 2006; Chambers et al., 2007; Bethard et al., 2007; Verhagen and Pustejovsky, 2008), the problem was formulated as a classification problem for determining the pair-wise event temporal relations; while recent works (Do et al., 2012; Mirza and Tonelli, 2016; Ning et al., 2017, 2018) took advantage of utilizing structural constraints such as transitive properties of temporal relationships via ILP to achieve better results. Comparatively, the concept of event causality knowledge here is broader and more flexible. Any event causality relation gained from human experience could be represented and utilized in KnowSemLM; as shown in Sec. 4.2 that such knowledge can be both mined from corpus and written down declaratively.

Since we formulate the semantic sequence modeling problem as a language modeling issue, we also review recent neural language modeling literature. Bengio et al. (2003) introduced a model that learns word vector representations as part of a simple neural network architecture for language modeling. Collobert and Weston (2008) decoupled the word vector training from the downstream training objectives, which paved the way for Collobert et al. (2011) to use the full context of a word for learning the word representations. The skip-gram and continuous bag-of-words (CBOW) models of Mikolov et al. (2013) propose a simple single-layer architecture based on the inner product between two word vectors. Mnih and Kavukcuoglu (2013) also proposed closely-related vector log-bilinear models, vLBL and ivLBL, and Levy and Goldberg (2014) proposed explicit word embeddings based on a PPMI metric. Additionally, researcher have been attempting to infuse knowledge into the language modeling process (Ahn et al., 2016; Yang et al., 2016; Ji et al., 2017; He et al., 2017; Clark et al., 2018).

Most recently, pre-trained language models such as BERT (Devlin et al., 2019), GPT (Radford et al., 2018), and XLNET (Yang et al., 2019) have achieved much success for language modeling and generation tasks. Our proposed knowledge infused semantic language model can not be directly applied upon such word-level pre-trained language models. However, as future works, we are interested in exploring the possibility of pre-training a semantic language model with frame and entity abstractions on a large corpus with event causality knowledge, and fine-tune it on application tasks.

7 Conclusion

This paper proposes KnowSemLM, a knowledge infused semantic LM. It utilizes both local context (i.e., what has been described in text) and global context (i.e., causality knowledge about events) to predict future events. We show that such event causality knowledge can be obtained statistically from a corpus or declaratively in specific scenarios. Similar to previous works, KnowSemLM takes advantage of event-level abstractions to achieve generalization. Evaluations demonstrate that the knowledge awareness of the proposed KnowSemLM helps improve results on tasks such as the event cloze test and story/referent prediction.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work was supported by the IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network, as well as by contracts HR0011-15-C-0113 and HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. 2016. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

David Bamman and Noah A. Smith. 2014. Unsupervised discovery of biographical structure from text. *Transactions of the Association for Computational Linguistics (TACL)*.

Cosmin Adrian Bejan. 2008. Unsupervised discovery of event scenarios from texts. In *Proceedings of the 21st International Florida Artificial Intelligence Research Society Conference*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research (JMLR)*.

Steven Bethard, James H Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *Proceedings of the International Conference on Semantic Computing (ICSC)*.

Susan Brown, Claire Bonial, Leo Obrst, and Martha Palmer. 2017. The rich event ontology. In *Proceedings of the Events and Stories in the News Workshop*.

Zheng Cai, Lifu Tu, and Kevin Gimpel. 2017. Pay attention to the ending: Strong neural baselines for the ROC story cloze task. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised Learning of Narrative Event Chains. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*.

Snigdha Chaturvedi, Haoruo Peng, and Dan Roth. 2017. Story comprehension for predicting what happens next. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic Frame Induction. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 2018. Neural text generation in stories using entity representations as context. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Quang Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Francis Ferraro and Benjamin Van Durme. 2016. A unified bayesian model of scripts, frames and language. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Mark Granroth-Wilding and Stephen Clark. 2016. What Happens Next? Event Prediction Using a Compositional Neural Network Model. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Naoya Inoue, Yuichiroh Matsubayashi, Masayuki Ono, Naoaki Okazaki, and Kentaro Inui. 2016. Modeling context-sensitive selectional preference with distributed representations. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Joseph Irwin, Mamoru Komachi, and Yuji Matsumoto. 2011. Narrative schema as world knowledge for coreference resolution. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Bram Jans, Steven Bethard, Ivan Vulic, and Marie-Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Daniel Khashabi, Mark Sammons, Ben Zhou, Tom Redman, Christos Christodoulopoulos, Vivek Srikumar, Nicholas Rizzolo, Lev Ratinov, Guanheng Luo, Quang Do, Chen-Tse Tsai, Subhro Roy, Stephen Mayhew, Zhili Feng, John Wieting, Xiaodong Yu, Yangqiu Song, Shashank Gupta, Shyam Upadhyay, Naveen Arivazhagan, Qiang Ning, Shaoshi Ling, and Dan Roth. 2018. CogCompNLP: Your Swiss Army Knife for NLP. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*.

I-Ta Lee and Dan Goldwasser. 2019. Multi-relational script learning for discourse relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.

Paramita Mirza and Sara Tonelli. 2016. CATENA: Causal and temporal relation extraction from natural language texts. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Ashutosh Modi and Ivan Titov. 2014a. Inducing neural models of script knowledge. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Ashutosh Modi and Ivan Titov. 2014b. Learning semantic script knowledge with event embeddings. In *ICLR Workshop*.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. 2017. Modeling semantic expectation: Using script knowledge for referent prediction. *Transactions of the Association for Computational Linguistics (TACL)*.

Raymond J. Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. 2017. LSDSem 2017 Shared Task: The Story Cloze Test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*.

Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M.

Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. 2018. Improving Temporal Relation Extraction with a Globally Acquired Statistical Resource. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Haoruo Peng, Kai-Wei Chang, and Dan Roth. 2015a. A joint framework for coreference resolution and mention head detection. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Haoruo Peng, Snigdha Chaturvedi, and Dan Roth. 2017. A joint model for semantic sequences: Frames, entities, sentiments. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015b. Solving hard coreference problems. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Haoruo Peng, Yangqiu Song, and Dan Roth. 2016. Event detection and co-reference with minimal supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Karl Pichotta and Raymond J. Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Karl Pichotta and Raymond J. Mooney. 2016a. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Karl Pichotta and Raymond J Mooney. 2016b. Using sentence-level lstm language models for script inference. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning causality for news events prediction. In *Proceedings of the International World Wide Web Conferences (WWW)*.

Kira Radinsky and Eric Horvitz. 2013. Mining the web to predict future events. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*.

Altaf Rahman and Vincent Ng. 2011. Coreference resolution with world knowledge. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Elena Rishes, Stephanie M Lukin, David K Elson, and Marilyn A Walker. 2013. Generating different story tellings from semantic representations of narrative. In *International Conference on Interactive Digital Storytelling*, pages 192–204. Springer.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Oxford, England: Lawrence Erlbaum.

Yangqiu Song, Haoruo Peng, Parisa Kordjamshidi, Mark Sammons, and Dan Roth. 2015. Improving a pipeline architecture for shallow discourse parsing. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Conference on Neural Information Processing Systems (NeurIPS)*.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the TARSQI toolkit. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Zhongqing Wang, Yue Zhang, and Ching-Yun Chang. 2017. Integrating order information and event relation for script event prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2016. Reference-aware language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Neural Attentive Bag-of-Entities Model for Text Classification

Ikuya Yamada[1,3]
ikuya@ousia.jp

Hiroyuki Shindo[2,3]
shindo@is.naist.jp

[1]Studio Ousia, Tokyo, Japan
[2]Nara Institute of Science and Technology, Nara, Japan
[3]RIKEN AIP, Tokyo, Japan

Abstract

This study proposes a *Neural Attentive Bag-of-Entities* model, which is a neural network model that performs text classification using entities in a knowledge base. Entities provide unambiguous and relevant semantic signals that are beneficial for capturing semantics in texts. We combine simple high-recall entity detection based on a dictionary, to detect entities in a document, with a novel neural attention mechanism that enables the model to focus on a small number of unambiguous and relevant entities. We tested the effectiveness of our model using two standard text classification datasets (i.e., the 20 Newsgroups and R8 datasets) and a popular factoid question answering dataset based on a trivia quiz game. As a result, our model achieved state-of-the-art results on all datasets. The source code of the proposed model is available online at https://github.com/wikipedia2vec/wikipedia2vec.

1 Introduction

Text classification is an important task, and its applications span a wide range of activities such as topic classification, spam detection, and sentiment classification. Recent studies showed that models based on neural networks can outperform conventional models (e.g., naïve Bayes) on text classification tasks (Kim, 2014; Iyyer et al., 2015; Tang et al., 2015; Dai and Le, 2015; Jin et al., 2016; Joulin et al., 2017; Shen et al., 2018). Typical neural network-based text classification models are based on words. They typically use words in the target documents as inputs, map words into continuous vectors (embeddings), and capture the semantics in documents by using compositional functions over word embeddings such as averaging or summation of word embeddings, convolutional neural networks (CNN), and recurrent neural networks (RNN).

Apart from the aforementioned approaches, past studies attempted to use entities in a knowledge base (KB) (e.g., Wikipedia) to capture the semantics in documents. These models typically represent a document by using a set of entities (or *bag of entities*) relevant to the document (Gabrilovich and Markovitch, 2006, 2007; Xiong et al., 2016). The main benefit of using entities instead of words is that unlike words, entities provide unambiguous semantic signals because they are uniquely identified in a KB. One key issue here is to determine the way in which to associate a document with its relevant entities. An existing straightforward approach (Peng et al., 2016; Xiong et al., 2016) involves creating a set of relevant entities using an entity linking system to detect and disambiguate the names of entities in a document. However, this approach is problematic because (1) entity linking systems produce disambiguation errors (Cornolti et al., 2013), and (2) entities appearing in a document are not necessarily relevant to the given document (Gamon et al., 2013; Dunietz and Gillick, 2014).

This study proposes the *Neural Attentive Bag-of-Entities* (NABoE) model, which is a neural network model that addresses the text classification problem by modeling the semantics in the target documents using entities in the KB. For each entity name in a document (e.g., *"Apple"*), our model first detects entities that may be referred to by this name (e.g., *Apple Inc.*, *Apple (food)*), and then represents the document using the weighted average of the embeddings of these entities. The weights are computed using a novel neural attention mechanism that enables the model to focus on a small subset of the entities that are less ambiguous in meaning and more relevant to the document. In other words, the attention mechanism is designed to compute weights by jointly addressing entity linking and entity salience detection (Ga-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 563–573
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

mon et al., 2013; Dunietz and Gillick, 2014) tasks. Furthermore, the attention mechanism improves the interpretability of the model because it enables us to inspect the small number of entities that strongly affect the classification decisions.

We validate the effectiveness of our proposed model by addressing two important natural language tasks: a text classification task using two standard datasets (i.e., the 20 Newsgroups and R8 datasets), and a factoid question answering task based on a popular dataset derived from the *quiz bowl* trivia quiz game. As a result, our model achieved state-of-the-art results on both tasks. The source code of the proposed model is available online at https://github.com/wikipedia2vec/wikipedia2vec.

2 Our Approach

Given a document, our model addresses the text classification task by using the following two steps: it first detects entities from the document, and then classifies the document using the proposed model with the detected entities as inputs.

2.1 Entity Detection

In this step, we detect entities that may be relevant to the document. Here, we use a simple method based on an *entity dictionary* that maps an entity name (e.g., "Washington") to a set of possible referent entities (e.g., *Washington, D.C.* and *George Washington*). In particular, we first take all words and phrases in a document, treat them as entity names if they exist in the dictionary, and detect all possible referent entities for each detected entity name. Following past work (Hasibi et al., 2016; Xiong et al., 2016), the boundary overlaps of the names are resolved by detecting only those that are the earliest and the longest.

We use Wikipedia as the target KB, and the entity dictionary is built by using the names and their referent entities of all internal anchor links in Wikipedia (Guo et al., 2013). We also collect two statistics from Wikipedia, namely *link probability* and *commonness* (Mihalcea and Csomai, 2007; Milne and Witten, 2008). The former is the probability of a name being used as an anchor link in Wikipedia, whereas the latter is the probability of a name referring to an entity in Wikipedia.

We generate a list of entities by concatenating all possible referent entities contained in the dictionary for each detected entity name, and feed it to the model presented in the next section. Note that we do not disambiguate entity names here, but detect all possible referent entities of the entity names.

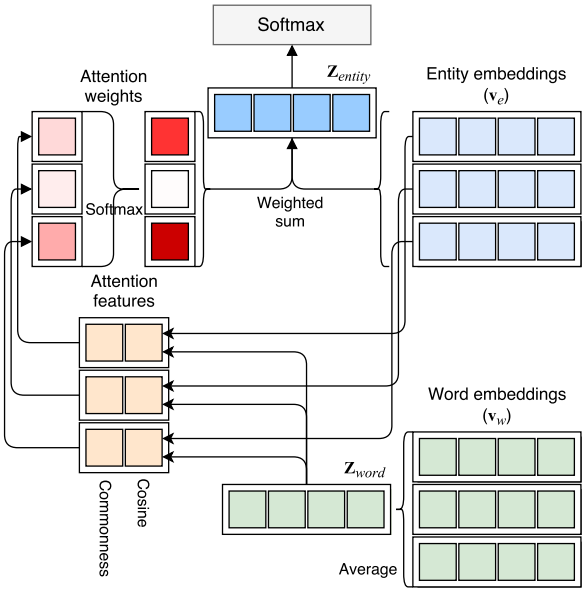


Figure 1: Architecture of the NABoE-entity model.

2.2 Model

Figure 1 shows the architecture of our model. Given words $w_1, ..., w_N$, and entities $e_1, ..., e_K$ detected from target document D, we first compute the word-based representation of D:

$$\mathbf{z}_{word} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{v}_{w_i}, \quad (1)$$

where $\mathbf{v}_w \in \mathbb{R}^d$ is the embedding of word w. We then derive the entity-based representation of D as a weighted average of the embeddings of the entities:

$$\mathbf{z}_{entity} = \sum_{i=1}^{K} a_{e_i} \mathbf{v}_{e_i}, \quad (2)$$

where $\mathbf{v}_e \in \mathbb{R}^d$ is the embedding of entity e and a_e the normalized attention weight corresponding to e computed using the following softmax-based attention function:

$$a_e = \frac{\exp(\mathbf{w}_a^\top \Phi(e, D) + b_a)}{\sum_{i=1}^{K} \exp(\mathbf{w}_a^\top \Phi(e_i, D) + b_a)}, \quad (3)$$

where $\mathbf{w}_a \in \mathbb{R}^l$ is a weight vector, $b_a \in \mathbb{R}$ is the bias, and $\Phi(e, D)$ is a function that generates an l-dimensional vector consisting of the features of the attention function.

We use the following two features in the attention function:

- **Cosine**: the cosine similarity between the embedding of the entity $\mathbf{v}_e$ and the word-based representation of the document $\mathbf{z}_{word}$.
- **Commonness**: the probability that the entity name refers to the entity in KB.

Here, our aim is to capture the relevance and the unambiguity of entity e in document D using the attention function. Thus, the problem is related to the tasks of entity salience detection (Gamon et al., 2013; Dunietz and Gillick, 2014), which aims to detect entities relevant (or salient) to the document, and entity linking, which aims to resolve the ambiguity of entities. The key assumption relating to these two tasks in the literature is that if an entity is semantically related to the given document, it is relevant to the document (Dunietz and Gillick, 2014), and it is likely to appear in the document (Milne and Witten, 2008; Ratinov et al., 2011). With this in mind and following past work (Yamada et al., 2016), we use the cosine similarity between $\mathbf{v}_e$ and $\mathbf{z}_{word}$ as a feature. Further, as in past entity linking studies, we also use the commonness of the name referring to the entity.

Moreover, we derive a representation based both on entities and words by simply adding $\mathbf{z}_{entity}$ and $\mathbf{z}_{word}$[1]:

$$\mathbf{z}_{full} = \mathbf{z}_{entity} + \mathbf{z}_{word}. \quad (4)$$

We then solve the task using a multiclass logistic regression classifier with the computed representation (i.e., with $\mathbf{z}_{entity}$ or $\mathbf{z}_{full}$) as features. In the remainder of this paper, we denote our models based on $\mathbf{z}_{entity}$ and $\mathbf{z}_{full}$ by **NABoE-entity** and **NABoE-full**, respectively.

3 Experimental Setup

In this section, we describe our experimental setup used both in the text classification and the factoid question answering experiments presented below.

3.1 Entity Detection

As the target KB, we used the September 2018 version of Wikipedia, which contains a total of 7,333,679 entities.[2] Regarding the entity dictionary described in Section 2.1, we excluded an entity name if its link probability was lower than 1% and a referent entity if its commonness given the entity name was lower than 3% for computational efficiency. Entity names were treated as case-insensitive. As a result, the dictionary contained 18,785,550 entity names, and each name had 1.14 referent entities on average.

Furthermore, to detect entities from a document, we also tested two publicly available entity linking systems, **Wikifier** (Ratinov et al., 2011; Cheng and Roth, 2013) and **TAGME** (Ferragina and Scaiella, 2012), instead of using dictionary-based entity detection.[3] We selected these systems because they are capable of detecting non-named entities (e.g., technical terms) that are useful for addressing the text classification task.[4] Here, we used the entities detected and disambiguated by these systems as inputs to our neural network model.

3.2 Pretrained Embeddings

We initialized the embeddings of words ($\mathbf{v}_w$) and entities ($\mathbf{v}_e$) using pretrained embeddings trained on KB. To learn embeddings from the KB, we used the method adopted in the open source Wikipedia2Vec tool (Yamada et al., 2016, 2018a). In particular, we generated an entity-annotated corpus from Wikipedia by treating entity links in Wikipedia articles as entity annotations, and trained skip-gram embeddings (Mikolov et al., 2013a,b) of 300 dimensions with negative sampling using the generated corpus as inputs. The learned embeddings place similar words and entities close to one another in a unified vector space. Here, we used the same version of Wikipedia described in Section 3.1.

4 Text Classification

To evaluate the effectiveness of our proposed model, we first conducted the text classification

[1] We also tested concatenating $\mathbf{z}_{entity}$ and $\mathbf{z}_{word}$ to derive $\mathbf{z}_{full}$; however, adding them generally achieved enhanced performance in our experiments presented below.

[2] We downloaded the Wikipedia dump from Wikimedia Downloads: https://dumps.wikimedia.org/

[3] In our experiments, we simply used all entities detected by the entity linking systems.

[4] In our preliminary experiments, we also tested three other state-of-the-art entity linking systems: AIDA (Hoffart et al., 2011), WAT (Piccinno and Ferragina, 2014), and the commercial Entity Analysis API in Google's Cloud Language service. However, these systems achieved lower overall performance compared to Wikifier and TAGME because they tended to ignore non-named entities.

task on two standard datasets, namely the 20 Newsgroups (20NG) (Lang, 1995) and R8 datasets (Debole and Sebastiani, 2005).

4.1 Setup

Our experimental setup described in this section follows that in past work (Liu et al., 2015; Jin et al., 2016; Yamada et al., 2018b). In particular, we used the 20NG and R8 datasets to train and test the proposed model. The 20NG dataset was created using the documents obtained from 20 Newsgroups and contained 11,314 training documents and 7,532 test documents.[5] The R8 dataset consisted of news documents from the eight most popular classes of the Reuters-21578 corpus (Lewis, 1992) and comprised 5,485 training documents and 2,189 test documents. We created the development set for each dataset by selecting 5% of the documents for training. Note that the class distribution of the R8 dataset is highly imbalanced. For example, the number of documents in the largest and smallest classes is 3,923 documents and 51 documents, respectively.

We report the accuracy and macro-average F1 scores. The model was trained using mini-batch stochastic gradient descent (SGD) with its batch size set to 32 and its learning rate controlled by Adam (Kingma and Ba, 2014). We used words and entities that were detected three times or more in the dataset and ignored the other words and entities. The size of the embeddings of words and entities was set to $d = 300$. We used early stopping based on the accuracy of the development set of each dataset to avoid overfitting of the model.

4.2 Baselines

We used the following models as our baselines:

- **BoW-SVM** (Jin et al., 2016): This model is based on a conventional linear support vector machine (SVM) with bag of words (BoW) features. It outperformed the conventional naïve Bayes-based model.

- **BoE** (Jin et al., 2016): This model extends the skip-gram model; It learns different word embeddings per target class from the dataset, and a linear model based on learned word embeddings is used to classify the documents. The performance of this model was superior to that of many state-of-the-art models, including those based on the skip-gram and CBOW models (Mikolov et al., 2013b), and the paragraph vector model (Le and Mikolov, 2014).

- **SWEM-concat** (Shen et al., 2018): This model is based on a neural network model with simple pooling operations (i.e., average and max pooling) over pretrained word embeddings.[6] Despite its simplicity, it outperformed many neural network-based models such as the word-based CNN model (Kim, 2014) and RNN model with LSTM units (Shen et al., 2018).

- **TextEnt** (Yamada et al., 2018b): This model learns entity-aware document embeddings from Wikipedia, and uses a neural network model with the learned embeddings as pretrained parameters to address text classification.

As described in Section 2.1, we also tested the variants of our NABoE-entity and NABoE-full models for which **Wikifier** and **TAGME** were used as the entity detection methods.

4.3 Results

Table 1 shows the results of our models and those of our baselines. Here, *w/o att.* and *w/o emb.* signify the model without the neural attention mechanism (all attention weights a_e are set to $\frac{1}{K}$, where K is the number of entities in the document) and the model without the pretrained embeddings (the embeddings are initialized randomly), respectively.

Relative to the baselines, our models yielded enhanced overall performance on both datasets. The NABoE-full model outperformed all baseline models in terms of both measures on both datasets. Furthermore, the NABoE-entity model outperformed all the baseline models in terms of both measures on the 20NG dataset, and the F1 score on the R8 dataset. Moreover, our attention mechanism consistently improved the performance. These results clearly highlighted the effectiveness of our approach, which addresses text

[5] We used the by-date version downloaded from the author's web site: http://qwone.com/~jason/20Newsgroups/.

[6] We also tested all four models proposed in Shen et al. (2018) (i.e., SWEM-aver, SWEM-max, SWEM-concat, and SWEM-hier). These models generally delivered comparable performance, with SWEM-concat slightly outperforming the other models on average.

	20NG		R8	
	Acc.	F1	Acc.	F1
NABoE-entity	.863	.856	.962	.915
NABoE-entity w/o att.	.822	.817	.943	.869
NABoE-entity w/o emb.	.844	.838	.957	.892
NABoE-full	**.868**	**.862**	**.971**	**.917**
Wikifier (NABoE-entity)	.735	.729	.896	.803
Wikifier (NABoE-entity w/o att.)	.728	.723	.844	.782
Wikifier (NABoE-entity w/o emb.)	.727	.722	.861	.755
Wikifier (NABoE-full)	.797	.789	.953	.839
TAGME (NABoE-entity)	.844	.838	.942	.871
TAGME (NABoE-entity w/o att.)	.826	.821	.924	.857
TAGME (NABoE-entity w/o emb.)	.842	.836	.942	.865
TAGME (NABoE-full)	.860	.853	.958	.889
BoW-SVM	.790	.783	.947	.851
BoE	.831	.827	.965	.886
SWEM-concat	.853	.855	.967	.898
TextEnt	.845	.839	.967	.910

Table 1: Results of the text classification task on the 20NG and R8 datasets. Here, *w/o att.* and *w/o emb.* represent the model without the neural attention mechanism and the model without the pretrained embeddings, respectively.

classification by using a small number of unambiguous and relevant entities detected by the proposed attention mechanism. Moreover, the pretrained embeddings improved the performance on both datasets.

Further, the models based on the dictionary-based entity detection (see Section 2.1) generally outperformed the models based on the entity linking systems (i.e., Wikifier and TAGME). We consider that this is because these entity linking systems failed to detect or disambiguate entity names that were useful to address the text classification task. Moreover, our attention mechanism consistently improved the performance for Wikifier- and TAGME-based models because the attention mechanism enabled the model to focus on entities that were relevant to the document.

4.4 Analysis

In this section, we provide a detailed analysis of the performance of our model in terms of conducting the text classification task. We first provide a comparison of the SWEM-concat, NABoE-entity, and NABoE-full models using class-level F1 scores on both of the datasets (see Table 2). Here, we aim to compare the detailed performance of the word-based model (SWEM-concat), entity-based model (NABoE-entity), and the model based on both words and entities (NABoE-full). Compared with the SWEM-concat model, the NABoE-full and NABoE-entity models performed more accurately in 23 out of 28 and 17 out of 28 classes, respectively. This result clearly demonstrates the ability of the model to successfully capture strong semantic signals that can only be obtained from entities. Moreover, we observed that the NABoE-entity model achieved weaker performance especially for the *misc.forsale* class in the 20NG dataset and several classes in the R8 dataset. Regarding the *misc.forsale* class, because documents in this class contain a wider variety of entities (i.e., objects users want to sell) than other classes, the model failed to capture the effective semantic signals from the entities. Further, as described in the error analysis provided below, it often appeared to be difficult to distinguish pairs of

Class	SWEM -concat	NABoE -full	NABoE -entity
20NG:			
alt.atheism	.780	**.820**	.804
comp.graphics	.787	.818	**.822**
comp.os.ms-windows.misc	.746	.802	**.811**
comp.sys.ibm.pc.hardware	.735	**.754**	.752
comp.sys.mac.hardware	.857	**.865**	.861
comp.windows.x	.837	.867	**.870**
misc.forsale	**.854**	.834	.805
rec.autos	.916	**.929**	.917
rec.motorcycles	.954	**.968**	.956
rec.sport.baseball	.946	**.969**	.966
rec.sport.hockey	.971	**.981**	.975
sci.crypt	**.942**	.940	.940
sci.electronics	.794	**.806**	.783
sci.med	.878	.900	**.905**
sci.space	.921	**.923**	.918
soc.religion.christian	.905	**.906**	.905
talk.politics.guns	.826	**.828**	.819
talk.politics.mideast	.921	**.940**	.935
talk.politics.misc	.689	**.694**	.680
talk.religion.misc	.657	.702	**.706**
R8:			
grain	.750	**.889**	**.889**
ship	.781	.817	**.822**
interest	.910	.885	.885
money-fx	**.909**	.894	.898
trade	.894	**.924**	**.924**
crude	**.971**	.958	.954
acq	.979	**.980**	.966
earn	.989	**.990**	.980

Table 2: Class-level F1 scores in each class on the 20NG and R8 datasets.

	20NG		R8	
	Acc.	F1	Acc.	F1
Commonness only	.849	.843	.949	.894
Cosine only	.846	.840	.956	.898
Both	**.863**	**.856**	**.962**	**.915**

Table 3: Feature study of the neural attention mechanism of the NABoE-entity model.

Class	Top entities
20NG:	
alt.atheism	Christian ethics, Atheism, Moral agency, Gregg Jaeger, Fred Rice
comp.graphics	Algorithm, Ray tracing (graphics), Framebuffer, Image file formats, TIFF
comp.os.ms-windows.misc	Windows 3.1x, Microsoft Windows, Windows NT, CONFIG.SYS, BMP file format
comp.sys.ibm.pc.hardware	BIOS, Don't Copy That Floppy, SCSI host adapter, Nonvolatile BIOS memory, Parallel SCSI
comp.sys.mac.hardware	PowerBook, Macintosh Quadra 610, Macintosh Quadra 650, FirstClass, Macintosh SE/30
comp.windows.x	X-Perts, Xterm, OPEN LOOK, OpenWindows, Man page
misc.forsale	Freight transport, Make Me an Offer, AC adapter, Plaque reduction neutralization test, Outline of working time and conditions
rec.autos	Manual Shift, Chassis, Automotive industry, Nissan, Ford Probe
rec.motorcycles	United States Department of Defense, Motorcycle, ZX8302, Honda motorcycles, Pillion, Hawk GT
rec.sport.baseball	Pitcher, Inning, The Jays, Home run, Bullpen
rec.sport.hockey	National Hockey League, Goaltender, ESPN, The Penguins, Achkar
sci.crypt	Cryptography, Algorithm, Escrow, Considered harmful, Encryption
sci.electronics	Solvent, Copy protection, Electronics, Lead–acid battery, Printed circuit board
sci.med	Infection, Antibiotics, Kirlian photography, Allergy, Kirlian
sci.space	Spacecraft, SunOS, Vandalism, VIA International, Space station
soc.religion.christian	Rutgers University, Geneva, Byler, Immaculate Conception, Original sin
talk.politics.guns	Ranch, BD's Mongolian Grill, Firearm, Second Amendment to the United States Constitution, Feustel
talk.politics.mideast	Serdar Argic, Israelis, Palestinians, Palestine Liberation Organization, Arabs
talk.politics.misc	Clayton Cramer, Janet Reno, Police state, Ronzone, Federal Bureau of Investigation
talk.religion.misc	Christian ethics, Thomas George Lanphier, David Koresh, Albert Sabin, Josephus
R8:	
grain	Grain, Tonne, Price support, Oil reserves, United States Senate
ship	Freight transport, Shipbuilding, Flag of convenience, Cargo, Persian Gulf
trade	Balance of trade, Export, International trade, Economic sanctions, Import
interest	Interest rate, Prime rate, Repurchase agreement, Balance of trade, Money market
money-fx	Exchange rate, Currency, Money market, Foreign exchange market, Monetary policy
crude	Petroleum, West Texas Intermediate, Price of oil, OPEC, Oil platform
acq	Common stock, Tender offer, Privately held company, Preferred stock, Shares outstanding
earn	QTR, Dividend, Stock split, Net profit, Income fund

Table 4: Top five influential entities for each class of the NABoE-entity model in the 20NG and R8 datasets.

similar classes in the R8 dataset based only on entities.

Next, we conducted a feature study of the attention mechanism by excluding one feature at a time from the NABoE-entity model (Table 3). We found both of the features to make an important contribution to the performance.

Furthermore, to investigate the attention mechanism in more detail, we computed the top influential entities in the attention mechanism for each class on the 20NG and R8 datasets. In particular, we calculated the number of times each entity obtained the highest attention weight in the test documents in each class and selected the five most frequent ones. Table 4 presents the results. Overall, our attention mechanism successfully selected entities that were highly relevant to each class. For example, *Cryptography*, *Algorithm*, *Escrow*, *Considered harmful*, and *Encryption* were selected for the *sci.crypt* class. Furthermore, although we did not explicitly perform entity disambiguation, the model successfully overcame the ambiguity issues in the entity names and attended to the entities that were relevant to the classes.

Subsequently, we conducted an error analysis by selecting 50 random test documents for which the NABoE-entity model made wrong predictions. Most of the errors were caused by two pairs of classes: 22 errors were caused by misclassifying documents of *acq* (corporate acquisitions) and those of *earn* (corporate earnings), and 13 errors were caused by misclassifying documents of *interest* and those of *money-fx*. Furthermore, the model tended to perform poorly if a document contained entities that strongly indicate an incorrect class. For example, a *money-fx* document containing the entity *interest rate* multiple times was classified into the *interest* class, and a document in the *acq* class reporting news related to oil companies (i.e., ExxonMobil and ZENEX) was classified into the *crude* class.

5 Factoid Question Answering

In this section, we address factoid question answering based on a dataset consisting of questions of the *quiz bowl* trivia quiz game. Factoid ques-

tion answering is one of the common settings of question answering that aims to predict an entity (e.g., events, authors, and books) that is described in a given question. The players of quiz bowl solve questions consisting of sentences that describe an entity. Quiz bowl questions have frequently been used for evaluating neural network-based models in recent studies (Iyyer et al., 2014, 2015; Yamada et al., 2017).

This task has a significantly larger number of target classes compared to the task addressed in the previous experiment. Our main aim here is to evaluate the effectiveness of using entities to capture the finer-grained semantics required to perform the task of factoid question answering effectively.

5.1 Setup

Our experimental setup described in this section follows that in past work (Xu and Li, 2016; Yamada et al., 2017). We address this task as a text classification problem that selects the most relevant answer from the possible answers observed in the dataset. We obtained the dataset proposed in Iyyer et al. (2014)[7]. We only used questions in the history and literature categories. Furthermore, we excluded questions of which the answers appear fewer than six times in the dataset. As a result, the number of candidate answers was 303 and 424 in the history and literature categories, respectively. We used 20% of questions each for the development set and test sets, and the remaining 60% for the training set. As a result, the training, development, and test sets consisted of 1,535, 511, and 511 questions for the history category, and 2,524, 840, and 840 questions for the literature category.

The settings we used to train the model were the same as those in the previous experiment (see Section 4.1). The model was trained using mini-batch SGD with its learning rate controlled by Adam (Kingma and Ba, 2014) and its mini-batch size set to 32. We used words and entities that were detected three times or more in the dataset, and ignored the other words and entities. The size of the embeddings of words and entities was set to $d = 300$. As in past work, we report the accuracy score, and the score on the development set was used for early stopping.

[7]This dataset was downloaded from the authors' web page: https://cs.umd.edu/~miyyer/qblearn/.

Name	History	Literature
NABoE-full	**.949**	**.985**
NABoE-entity	.941	.979
NABoE-entity w/o att.	.845	.943
NABoE-entity w/o emb.	.941	.973
Wikifier (NABoE-full)	.935	.967
Wikifier (NABoE-entity)	.930	.952
Wikifier (NABoE-entity w/o att.)	.924	.941
Wikifier (NABoE-entity w/o emb.)	.934	.949
TAGME (NABoE-full)	.941	.977
TAGME (NABoE-entity)	.930	.963
TAGME (NABoE-entity w/o att.)	.922	.961
TAGME (NABoE-entity w/o emb.)	.932	.962
BoW	.508	.462
FTS-BRNN	.881	.931
NTEE	.947	.951
SWEM-concat	.900	.966

Table 5: Accuracy of the proposed and baseline methods for the factoid QA task.

5.2 Baselines

We used the following baseline models:

- **BoW** (Xu and Li, 2016) This model is based on a logistic regression classifier with conventional binary BoW features.
- **FTS-BRNN** (Xu and Li, 2016) This model is based on a bidirectional RNN with gated recurrent units (GRU). It uses the logistic regression classifier with the features derived by the RNN.
- **NTEE** (Yamada et al., 2017) This model is a state-of-the-art model that uses a multi-layer perceptron classifier with the features computed using the embeddings of words and entities trained on Wikipedia using the neural network model proposed in their paper.

Similar to our previous experiment, we also add **SWEM-concat**, and the variants of our NABoE-entity and NABoE-full models based on **Wikifier** and **TAGME** (see Section 4.2). Note that all the baselines address the task as a text classification problem.

5.3 Results and Analysis

Table 5 provides the results of our models and those of our baselines. Overall, our models achieved enhanced performance on this task. In particular, the NABoE-full model successfully outperformed all the baseline models, and the NABoE-entity model achieved competitive performance and outperformed all the baseline models in the literature category. These results clearly

highlighted the effectiveness of our model for this task.

Furthermore, similar to the previous text classification experiment, the attention mechanism and the pretrained embeddings consistently improved the performance. Moreover, the models based on dictionary-based entity detection outperformed the models based on the entity linking systems.

We also conducted an error analysis using the NABoE-entity model and the test questions in the history category. We found nearly 70% of the errors to be caused by questions of which the answers were country names. This is because these questions tended to provide indirect clues (e.g., describing a notable person born in the country) and most entities used in these clues do not directly indicate the answer (i.e., country names). Furthermore, our model failed in difficult cases such as predicting *Tokugawa shogunate* instead of *Tokugawa Ieyasu*.

6 Related Work

KB entities have been conventionally used to model the semantics in texts. A representative example is Explicit Semantic Analysis (ESA) (Gabrilovich and Markovitch, 2006, 2007), which represents a document using a bag of entities, namely a sparse vector of which each dimension corresponds to the relevance score of the text to each entity. This simple method is shown to be effective for various NLP tasks including text classification (Gabrilovich and Markovitch, 2006; Gupta and Ratinov, 2008; Negi and Rosner, 2013) and information retrieval (Egozi et al., 2011; Xiong et al., 2016),

Several neural network models that use KB entities to capture the semantics in texts have been proposed. These models typically depend on an additional preprocessing step that extracts the relevant entities from the target texts. For example, Wang et al. (2017) used the Probase conceptualization API for short text classification by retrieving the Probase entities that were relevant to the target text and used them in a model based on CNN. Pilehvar et al. (2017) also extracted entities using a graph-based linking algorithm and used these entities in a neural network model. A similar approach was adopted in Yamada et al. (2018b,c); they extracted entities from the target text using an entity linking system and simply used the detected entities in a neural network model. However, unlike these models, our proposed model addresses the task in an *end-to-end* manner; i.e., entities that are relevant to the target text are automatically selected using our neural attention mechanism. Furthermore, we also used the model proposed by Yamada et al. (2018b) as a baseline in our text classification experiments.

Additionally, our work is also related to studies on entity linking. Entity linking models can be roughly classified into two groups: *local* models, which resolve entity names independently using the contextual relevance of the entity given a document, and *global* models, in which all the entity names in a document are resolved simultaneously to select a topically coherent set of results (Ratinov et al., 2011). Recent state-of-the-art models typically combine both of these models (Yamada et al., 2016; Ganea and Hofmann, 2017; Cao et al., 2018; Kolitsas et al., 2018). However, several studies also showed that the local model alone can achieve results competitive to those of the global and combined models (Eshel et al., 2017; Ganea and Hofmann, 2017; Yamada et al., 2017; Cao et al., 2018; Kolitsas et al., 2018). In this study, we adopt a simple but effective local model, which uses cosine similarity between the embedding of the target entity and the word-based representation of the document to capture the relevance of an entity given a document.

7 Conclusions

This study proposed NABoE, which is a neural network model that performs text classification using entities in Wikipedia. We combined simple dictionary-based entity detection with a neural attention mechanism to enable the model to focus on a small number of unambiguous and relevant entities in a document. We achieved state-of-the-art results on two important NLP tasks, namely text classification and factoid question answering, which clearly verified the effectiveness of our approach. As a future task, we intend to more extensively analyze our model and explore its effectiveness for other NLP tasks. Furthermore, we would also like to test more expressive neural network models for example by integrating global entity coherence information into our neural attention mechanism.

References

Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Neural Collective Entity Linking. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 675–686.

Xiao Cheng and Dan Roth. 2013. Relational Inference for Wikification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1787–1796.

Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. 2013. A Framework for Benchmarking Entity-annotation Systems. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 249–260.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised Sequence Learning. In *Advances in Neural Information Processing Systems 28*, pages 3079–3087.

Franca Debole and Fabrizio Sebastiani. 2005. An Analysis of the Relative Hardness of Reuters-21578 Subsets: Research Articles. *Journal of the American Society for Information Science and Technology*, 56(6):584–596.

Jesse Dunietz and Daniel Gillick. 2014. A New Entity Salience Task with Millions of Training Examples. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 205–209.

Ofer Egozi, Shaul Markovitch, and Evgeniy Gabrilovich. 2011. Concept-Based Information Retrieval Using Explicit Semantic Analysis. *ACM Trans. Inf. Syst.*, 29(2):8:1—-8:34.

Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. 2017. Named Entity Disambiguation for Noisy Text. In *Proceedings of the 21st Conference on Computational Natural Language Learning*, pages 58–68.

Paolo Ferragina and Ugo Scaiella. 2012. Fast and Accurate Annotation of Short Texts with Wikipedia Pages. *Software, IEEE*, 29(1):70–75.

Evgeniy Gabrilovich and Shaul Markovitch. 2006. Overcoming the Brittleness Bottleneck using Wikipedia: Enhancing Text Categorization with Encyclopedic Knowledge. In *Proceedings of the 21st National Conference on Artificial Intelligence*, volume 2, pages 1301–1306.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing Semantic Relatedness Using Wikipedia-Based Explicit Semantic Analysis. In *International Joint Conference on Artificial Intelligence*, pages 1606–1611.

Michael Gamon, Tae Yano, Xinying Song, Johnson Apacible, and Patrick Pantel. 2013. Identifying Salient Entities in Web Pages. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 2375–2380.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep Joint Entity Disambiguation with Local Neural Attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629.

Stephen Guo, Ming-Wei Chang, and Emre Kiciman. 2013. To Link or Not to Link? A Study on End-to-End Tweet Entity Linking. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1020–1030.

Rakesh Gupta and Lev Ratinov. 2008. Text Categorization with Knowledge Transfer from Heterogeneous Data Sources. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, pages 842–847.

Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. Exploiting Entity Linking in Queries for Entity Retrieval. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, pages 209–218.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A Neural Network for Factoid Question Answering over Paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 633–644.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691.

Peng Jin, Yue Zhang, Xingyuan Chen, and Yunqing Xia. 2016. Bag-of-embeddings for Text Classification. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2824–2830.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-End Neural Entity Linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529.

Ken Lang. 1995. NewsWeeder: Learning to Filter Netnews. *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339.

Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31th International Conference on Machine Learning*, volume 32, pages 1188–1196.

David D. Lewis. 1992. An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–50.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical Word Embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2418–2424.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, pages 233–242.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the 2013 International Conference on Learning Representations*, pages 1–12.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

David Milne and Ian H. Witten. 2008. Learning to Link with Wikipedia. In *Proceeding of the 17th ACM Conference on Information and Knowledge Management*, pages 509–518.

Sapna Negi and Michael Rosner. 2013. UoM: Using Explicit Semantic Analysis for Classifying Sentiments. In *Proceedings of the Seventh International Workshop on Semantic Evaluation*, pages 535–538.

Hao Peng, Jing Liu, and Chin-Yew Lin. 2016. News Citation Recommendation with Implicit and Explicit Semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 388–398.

Francesco Piccinno and Paolo Ferragina. 2014. From TagME to WAT: A New Entity Annotator. In *Proceedings of the First International Workshop on Entity Recognition and Disambiguation*, pages 55–62.

Mohammad Taher Pilehvar, Jose Camacho-Collados, Roberto Navigli, and Nigel Collier. 2017. Towards a Seamless Integration of Word Senses into Downstream NLP Applications. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1857–1869.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and Global Algorithms for Disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1375–1384.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1432.

Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2915–2921.

Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2016. Bag-of-Entities Representation for Ranking. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, pages 181–184.

Dong Xu and Wu-Jun Li. 2016. Full-Time Supervision based Bidirectional RNN for Factoid Question Answering. *arXiv preprint arXiv:1606.05854v2*.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2018a. Wikipedia2Vec: An Optimized Tool for Learning Embeddings from Wikipedia. *arXiv preprint arXiv:1812.06280v2*.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning Distributed Representations of Texts and Entities from Knowledge Base. *Transactions of the Association for Computational Linguistics*, 5:397–411.

Ikuya Yamada, Hiroyuki Shindo, and Yoshiyasu Takefuji. 2018b. Representation Learning of Entities and Documents from Knowledge Base Descriptions. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 190–201.

Ikuya Yamada, Ryuji Tamaki, Hiroyuki Shindo, and Yoshiyasu Takefuji. 2018c. Studio Ousia's Quiz Bowl Question Answering System. In *The NIPS '17 Competition: Building Intelligent Systems*, pages 181–194.

Roll Call Vote Prediction with Knowledge Augmented Models

Pallavi Patil[♦] **Kriti Myer**[♦] **Ronak Zala**[♦] **Arpit Singh**[♦]
Sheshera Mysore[♦] **Andrew McCallum**[♦] **Adrian Benton**[♠] **Amanda Stent**[♠]

[♦]College of Information and Computer Sciences
University of Massachusetts Amherst
[♠]Bloomberg LP
{ppatil,kmyer,rzala,arpitsingh,smysore,mccallum}@cs.umass.edu
{abenton10,astent}@bloomberg.net

Abstract

The official voting records of United States congresspeople are preserved as roll call votes. Prediction of voting behavior of politicians for whom no voting record exists, such as individuals running for office, is important for forecasting key political decisions. Prior work has relied on past votes cast to predict future votes, and thus fails to predict voting patterns for politicians without voting records. We address this by augmenting a prior state of the art model with multiple sources of external knowledge so as to enable prediction on unseen politicians. The sources of knowledge we use are news text and Freebase, a manually curated knowledge base. We propose augmentations based on unigram features for news text, and a knowledge base embedding method followed by a neural network composition for relations from Freebase. Empirical evaluation of these approaches indicate that the proposed models outperform the prior system for politicians with complete historical voting records by 1.0% point of accuracy (8.7% error reduction) and for politicians without voting records by 33.4% points of accuracy (66.7% error reduction). We also show that the knowledge base augmented approach outperforms the news text augmented approach by 4.2% points of accuracy.

1 Introduction

Roll call votes are official records of how politicians vote on bills (potential laws) in the United States House of Representatives and Senate. Reliable prediction of these votes, using historical voting records and the text of bills, can be used to forecast political decisions on key issues, which can be informative for the electorate and other political stakeholders. Prior work has used politicians' voting records as a means to study their ideological stances (Poole and Rosenthal, 1985; Clinton et al., 2004), as well as roll call votes combined with the text of the corresponding bills to predict votes on newly drafted bills (Gerrish and Blei, 2012; Kraft et al., 2016; Kornilova et al., 2018). However, these approaches fail to make good predictions for the votes of politicians whose records are not established, such as new candidates for office – a time when this information can be most useful for the electorate. We hypothesize that additional sources of knowledge about new politicians can predict their future votes.

In this work we explore two sources of additional knowledge about politicians to better predict roll call votes: news article text about the politicians, and Freebase (Bollacker et al., 2008), which is a manually curated knowledge base (KB). Relevant news articles may contain words that are indicative of a politician's stance with respect to specific issues. A KB such as Freebase is likely to contain rich information such as events a congressperson attends, people they are related to, and personal details such as schools they were educated at; this information may be correlated with a politician's stance on specific issues (Sunshine Hillygus, 2005; Duckitt and Sibley, 2010; Kraut and Lewis, 1975). Information in a KB is likely to be more restricted, but more reliable, than information extracted from news articles.

We integrate these sources of information into the embedding based prediction model proposed in Kraft et al. (2016). We experiment with two representations for news articles: as the mean of the embeddings of the words in the article, and as a bag of words. To represent information in KBs, we first capture the KB relations using Universal Schema (US) (Riedel et al., 2013), and then construct relation embeddings using a neural network.

We evaluate the proposed approaches on multiple sessions of Congress under two settings: (1) with only politicians that are observed at train-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 574–581
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

ing time, which is similar to the setting of prior work (Kraft et al., 2016; Kornilova et al., 2018) and (2) where a subset of politicians' voting patterns are never observed at training time, representing the new candidate for office setting. We establish a new state-of-the-art under the evaluation framework used by most prior work (Setting 1). We also show that our approach outperforms a state-of-the-art model in our evaluation framework (Setting 2). Compared to the previous state-of-the-art model for roll call prediction, in Setting (1) our best approach gives an improvement in accuracy of 1.0% (an error reduction of 8.7%), and in Setting (2) our best approach gives an improvement in accuracy of 33.4% (an error reduction of 66.7%). Additionally, in Setting (2), augmentation via KB gives 4.2% more accurate predictions than augmentation from news text.

Code to reproduce our experiments can be found here: https://github.com/ronakzala/universal-schema-bloomberg/.

2 Models

All the models we explore take as input the voting record (denoted $\mathcal{V}$) for a politician p, the text of a bill b, and (optional) knowledge about the politician (denoted $\mathcal{K}$), and output a probability $\mathrm{P}(y = \texttt{Yea}|b, \mathcal{V}, \mathcal{K})$ that politician p will vote Yea on bill b. In this discussion, all features (voting record and knowledge augmentation) as well as labels (vote predictions) are specific to p, but for ease of reading we don't subscript to p.

We start with a baseline model that does not use any additional knowledge. We augment this model with knowledge from news articles mentioning p (denoted $\mathcal{N}$) and from parts of the Freebase KB relevant to p (denoted $\mathcal{F}$) (i.e. $\mathcal{N} \subset \mathcal{K}$ and $\mathcal{F} \subset \mathcal{K}$). In our augmented models, vector representations of this additional knowledge are denoted as $\mathbf{v}_\mathcal{N}$ and $\mathbf{v}_\mathcal{F}$. These vectors can be read as complementary representations of politician p, derived from how the politician appears in news articles and in curated sources of world knowledge, rather than by prior voting behavior. These vectors are used in conjunction with the politician's representation in terms of their historical voting record, denoted as $\mathbf{v}_\mathcal{V}$, which is learned as a model parameter. Now, we describe the forms of the baseline and augmented models.

2.1 Baseline Model

Our baseline is the model proposed by Kraft et al. (2016). This model represents a politician p by a vector $\mathbf{v}_\mathcal{V}$ and a bill b using a bag of words of the 1,000 most frequent unigrams across all bills, after excluding stopwords and single-character tokens. The model is defined as follows:

$$\mathbf{v}_b = \sum_{w \in b} \mathbf{e}_w / |b|$$
$$\mathbf{Z}_b = \mathbf{W}_{bill} \cdot \mathbf{v}_b + \mathbf{d}_{bill}$$
$$\mathrm{P}(y = \texttt{yea}|b, \mathcal{V}) = \sigma(\mathbf{Z}_b \cdot \mathbf{v}_\mathcal{V})$$

Here, $e_w \in \mathbb{R}^{d_{word}}$ is initialized to pre-trained GloVe word embeddings[1] and finetuned as a model parameter. $\mathbf{v}_\mathcal{V} \in \mathbb{R}^{d_{emb}}$ is the embedding for politician p and is initialized uniformly at random in $[-10^{-2}, 10^{-2}]$. Bill b is represented as $\mathbf{v}_b$, the average of word embeddings $\mathbf{e}_w$, and then transformed into the same space as $\mathbf{v}_\mathcal{V}$ via weight matrix $W_{bill} \in \mathbb{R}^{d_{emb} \times d_{word}}$ and bias vector $d_{bill} \in \mathbb{R}^{d_{emb}}$, i.e. $\mathbf{Z}_b \in \mathbb{R}^{d_{emb}}$.

2.2 News Text Augmented Model

We incorporate knowledge about politician p from relevant news articles in the form of unigram features extracted from the set of news articles that mention p, denoted $\mathcal{N}$. This model represents each article $a_j \in \mathcal{N}$ as a bag of words of the most frequent 2,000 unigrams across all articles for all politicians, after excluding stopwords and single-character tokens. The news augmented model is formulated as:

$$\mathbf{v}_b = \sum_{w \in b} \mathbf{e}_w / |b|$$
$$\mathbf{Z}_b = \mathbf{W}_{bill} \cdot \mathbf{v}_b + \mathbf{d}_{bill}$$
$$\mathbf{Z}_\mathcal{N} = \mathbf{W}_{news} \cdot \mathbf{v}_\mathcal{N} + \mathbf{d}_{news}$$
$$\mathrm{P}(y = \texttt{yea}|b, \mathcal{V}, \mathcal{N}) = \sigma(\mathbf{Z}_b \cdot [\mathbf{v}_\mathcal{V} + \mathbf{Z}_\mathcal{N}])$$

Most of the notation above is same as the one for the baseline model in §2.1. Here, $\mathbf{v}_\mathcal{N}$ is a vector representing the knowledge about p contained in N. $\mathbf{Z}_\mathcal{N} \in \mathbb{R}^{d_{emb}}$ represents $\mathbf{v}_\mathcal{N}$ transformed into the space of $\mathbf{v}_\mathcal{V}$ via weight matrix $\mathbf{W}_{news}$ and bias vector d_{news}. We experiment with two variations for computing $\mathbf{v}_\mathcal{N}$. First we compute $\mathbf{v}_\mathcal{N}$ as the mean GloVe vectors of all unigrams in $\mathcal{N}$

[1] http://nlp.stanford.edu/data/glove.6B.zip

(model denoted by NWGL):

$$\mathbf{v}_{\mathcal{N}}^{\text{GloVe}} = \frac{1}{\sum_{a \in \mathcal{N}} |a|} \sum_{a \in \mathcal{N}} \sum_{w \in a} \mathbf{e}_w$$

where $\mathbf{e}_w$ represents the GloVe vector for word w in article a. We also consider a variant where each article $a \in \mathcal{N}$ is represented as a vector of relative frequencies of the words in a (model denoted by NWFR):

$$\mathbf{v}_{\mathcal{N}}^{\text{FREQ}} = \sum_{a \in \mathcal{N}} \mathbf{f}_a$$

where $\mathbf{f}_a$ is a vector of unigram relative frequencies in article $a \in \mathcal{N}$.

2.3 Knowledge Base Augmented Model

Building on the baseline model, the knowledge base augmented model (denoted by KBUS) represents the contextual information for politician p as a vector $\mathbf{v}_{\mathcal{F}}$. The KB augmented model is formulated as:

$$\mathbf{v}_b = \sum_{w \in b} \mathbf{e}_w / |b|$$
$$\mathbf{Z}_b = \mathbf{W}_{bill} \cdot \mathbf{v}_b + \mathbf{d}_{bill}$$
$$\text{P}(y = \texttt{yea} | b, \mathcal{V}, \mathcal{F}) = \sigma(\mathbf{Z}_b \cdot [\mathbf{v}_{\mathcal{V}} + \mathbf{Z}_{\mathcal{F}}])$$

Most of the notation above remains the same as the baseline model in §2.1. $\mathbf{Z}_{\mathcal{F}}$ is created from an embedded subgraph of the Freebase KB. Freebase consists of relation triples of the form (e_1, r, e_2), where e_1, and e_2 denote entities (e.g., `Barack Obama`, `Columbia University`) and r denotes a relation (e.g., `graduate_of`). These relation triples are embedded in a vector space by Universal Schema (Riedel et al., 2013), giving vector representations $\mathbf{v}_{e_1}$, $\mathbf{v}_{e_2}$, and $\mathbf{v}_r$ for the elements of the triple, e_1, e_2, and r. Universal Schema embeddings for entities and relations are trained for the KB completion task, to maximize the probability of triples existing in the KB parameterized by $\text{P}(r, e_1, e_2) = \sigma(\mathbf{v}_r^T [\mathbf{v}_{e_1}; \mathbf{v}_{e_2}])$.

The KB knowledge vector $\mathbf{Z}_{\mathcal{F}}$ is derived from the Universal Schema entity embeddings. For each politician p, we consider the subset of relations in which the politician is either e_1 or e_2, denoted $\mathcal{F}$. We link p to a KB entity in Freebase by exact textual match on p's name. Let r_i be a relation in a triple in $\mathcal{F}$ and o_i be the entity that is *not* politician p in that triple; we compute $\mathbf{Z}_{\mathcal{F}}$ as:

$$\mathbf{Z}_{\mathcal{F}} = \sum_{i \in \mathcal{F}} \mathbf{t}_i \cdot \text{P}(r_i, o_i, p)$$
$$\mathbf{t}_i = \text{FFN}([\mathbf{v}_{r_i}; \mathbf{v}_{o_i}])$$

Here $[;]$ denotes concatenation, v_{r_i} the Universal Schema embedding for relation r_i, and v_{o_i} the Universal Schema embedding for o_i. $\mathbf{t}_i$ is the embedding of the ith triple and is computed by passing v_{r_i} and v_{o_i} through a shallow feed forward network (FFN) of the form: $tanh(\mathbf{W}\mathbf{x} + \mathbf{b})$. The parameters of this FFN are learned in the course of model training, and the dimensionality of $\mathbf{t}_i$ is a hyperparameter. The final representation for the contextual knowledge from the KB for politician p is therefore a sum of the Freebase triple Universal Schema embeddings weighted by the probability that p would participate in each triple.

3 Experiment Description

We evaluate the proposed knowledge augmented models under two evaluation protocols (§3.3): one where all politicians' voting records are observed in the training data (Setting 1) and another in which some politicians are unobserved at training time, representing the new candidate for office setting (Setting 2). We also evaluate the performance of simple baseline methods and previous state-of-the-art methods (§3.2) in both settings. Finally, we perform a close comparison of baseline methods against the proposed methods and highlight individual politicians for which the knowledge augmented models better predict voting behavior than the baselines (§4.0.4).

3.1 Datasets

Our data comes from three sources: roll call voting records from Congress Sessions 106-109, news articles from the Concretely Annotated New York Times Corpus (Ferraro et al., 2014), and the KB from Freebase. Here we briefly describe each source and major preprocessing decisions.

3.1.1 Roll Call Votes

The roll call votes dataset was compiled to resemble the one used by Kraft et al. (2016). The dataset covers the 106th-109th Congress (1999-2006), where each session spans 2 years, and was created by querying GovTrack[2] using pub-

[2]Online Roll Call Votes: `https://www.govtrack.us/`

licly available tools[3]. Bill texts, bill metadata, and lists of roll call votes were all queried separately and matched using bill IDs. The resulting dataset contains voting records for 709 unique politicians across sessions and almost 1 million politician-bill-vote examples. In order to facilitate future research on this data we make our tools and raw data available to the community, a component absent from prior work[4].

3.1.2 New York Times Corpus

We use the The Concretely Annotated NYT Corpus (Ferraro et al., 2014) as a source of knowledge contained in news articles. This corpus contains approximately 1.8 million articles for the period 1987-2007, which includes the sessions of Congress for which we report experimental results. The corpus is automatically annotated using the CoreNLP package with a host of annotations such as part of speech and named entity tags.

To obtain the set of politician-relevant articles for the news article augmented model (§2.2), we first construct a list of politician names from our roll call vote data and Wikipedia aliases for these names. We identify candidate names in each article by extracting all spans in the article annotated with the `Person` entity type tag. We then look for exact string matches to the list of politician names in the candidate names extracted from each article. By this means, we identify a total of 50,800 relevant articles. Each article is represented as a bag of words from the 2,000 most frequent words across all articles after dropping stop words and single character words. When training and evaluating our models, we conservatively include only the subset of relevant articles published before the congressional session from which the bill comes; this way, information from news text necessarily predates the congressional votes and does not inform about the model about voting outcomes.

3.1.3 Freebase

We use Freebase as a source of KB knowledge. Freebase is a large, structured knowledge base that consists of relation triples created by human contributors. It contains about 46 million unique entities and 332 million relation triples.

In extracting relations relevant to politicians in our roll call vote data, we first filter to only relations that mention a politician explicitly as one of the entities. We add to this set relations one hop away from the politicians in the knowledge base (the politicians do not directly participate in this set of one hop away relations). The combination of direct and one hop away relations gives about 800,000 relation triples. This subset of Freebase is embedded with Universal Schema, following which only the direct politician relations, along with the scores learned by Universal Schema for these relations, are used in our KB augmented model (§2.3). Although Freebase relations are not temporally marked, Freebase does not contain politicians' voting records, so vote information cannot "bleed into" our models from Freebase.

3.2 Baselines

We compare the performance of the proposed models against several baseline and previous state-of-the-art models. These are:

KJR16: The model from Kraft et al. (2016). The numbers we report for KJR16 are based on our re-implementation of the original model. We also apply slightly different preprocessing decisions to the text of bills. For example, we drop stop words and single-character tokens, unlike the original paper.

MAJ: This baseline model predicts the majority class (`Yea`) for all votes on all bills.

PARTYM: This baseline predicts that a congressperson will vote for a bill in the direction of their party's majority vote for that bill. A congressperson can have a party affiliation of Republican, Democrat, or Independent. A politician's party is generally a very strong predictor of their voting behavior. We compare against this baseline specifically to measure how much gain contextual knowledge about a politician can bring over just predicting that a congressperson will toe the party line. This baseline operates in a slightly unrealistic setting, since the party majority vote is determined *after* all the votes have been cast for a given bill.

3.3 Experimental Setup

For all experiments, our models are trained and evaluated on each Congress session separately. This setup resembles that of most prior work. One exception is Kornilova et al. (2018), who evaluate

[3]Tools to query online congressional data sources: https://github.com/unitedstates/congress

[4]https://github.com/ronakzala/universal-schema-bloomberg/

Session	Majority Class	Accuracy (%)		Precision (%)		Recall (%)		F1 (%)	
		KJR16	BP	KJR16	BP	KJR16	BP	KJR16	BP
106	83.03	85.90	**86.49**	**90.46**	89.14	92.78	**95.32**	91.60	**92.13**
107	85.86	91.10	**91.63**	91.65	**92.57**	98.06	**98.51**	95.06	**95.44**
108	87.10	90.64	**91.68**	91.88	**93.64**	**97.27**	97.15	94.38	**95.36**
109	83.48	86.24	**88.07**	91.85	**92.28**	91.26	**93.05**	91.05	**92.66**
Average	84.87	88.47	**89.47**	91.46	**91.91**	94.84	**96.00**	93.02	**93.89**

Table 1: Performance of KJR16 (our reimplementation of Kraft et al. (2016)) compared to our Best Proposed (BP) model (News augmented-Glove, NWGL), in evaluation Setting (1).

on the next Congress session; even in this case, they restricted their evaluation sets to only those politicians that were observed at training time.

We use a train/dev/test split of $60\%/20\%/20\%$; every session of Congress contains about 250,000 politician-bill-vote examples given that each congress contains between 500-650 bills, and approximately 550 politicians. All of the reported numbers are averaged over four random restarts of the models.

As noted earlier, we evaluate the models under two settings: (Setting 1) in this setting all politicians are present in both training and test data (this is the setting used in all prior work); and (Setting 2) in this setting, votes from 5% of the politicians, chosen at random for each session of Congress, are removed from the training set[5], resulting in a reduction of around 7,000 politician-bill-vote examples from each session of Congress. All of these politicians are still present in the test set.

3.4 Model Hyperparameters and Training

All the models we train (KJR16, NWGL, NWFR, KBUS) use a bill embedding ($\mathbf{v}_b$) of size 50, and a per-politician embedding $\mathbf{v}_\mathcal{V}$ of size 10. The NWGL model additionally uses a mean GloVe vector ($\mathbf{v}_\mathcal{N}^{\text{GloVe}}$) of size 50, whereas the NWFR model uses a word frequency vector ($\mathbf{v}_\mathcal{N}^{\text{FREQ}}$) of size 2,000. The KBUS model has an entity embedding ($\mathbf{v}_o$) of size 25, and relation embedding ($\mathbf{v}_r$) of size 25, resulting in a KB knowledge vector ($\mathbf{Z}_\mathcal{F}$) of size 10 using the FFN architecture: $\{50, 10\}$. These settings were chosen without exploration; further hyperparameter tuning may result in different model performance.

All models were trained using vanilla SGD with a learning rate of 0.1 for up to 20 epochs[6]. We trained with early stopping based on the accuracy on the development set of politician-bill pairs.

For all models, we report accuracy, precision, recall, and F averaging over all vote predictions (i.e. we micro-average).

4 Results and Analysis

Our experiments attempt to answer several questions: (1) Setting 1: Does politician-related knowledge augmentation improve predictive performance when voting records of all politicians are observed (§4.0.1)? (2) Setting 2: Does politician-related knowledge augmentation improve performance when voting records of some politicians are not observed (§4.0.2)? (3) Does knowledge augmentation from a manually curated KB improve model performance compared to knowledge augmentation using unstructured text (§4.0.3)? (4) Finally, for which politicians are our knowledge augmented models more effective than predicting a party majority (PARTYM) vote (§4.0.4)?

4.0.1 Setting 1: All Voting Records Observed

Tables 1 and 2a display the performance of our proposed models when the voting patterns of all politicians are observed. Our best proposed model NWGL outperforms KJR16 in all sessions of Congress according to most metrics (Table 1). NWGL outperforms KJR16 by 1% point of accuracy on average, an error reduction of 8.7%. Knowledge augmentation in any form (NWGL, NWFR and KBUS) gives small improvements in roll call vote prediction over KJR16 (Table 2a).

4.0.2 Setting 2: Some Voting Records Absent

Table 2b displays results in the setting where some politicians' voting data was removed from the training set, representing the new candidate for office setting. For this setting, KJR16 makes random predictions. By contrast, all our models are able to

[5]On average, 2-10% of politicians are newcomers during every session, making our removal of 5% politicians realistic.

[6]No momentum or minibatching was used.

Model	Accuracy (%)	F1
MAJ	*83.82*	*91.19*
PARTYM	*83.56*	*90.54*
KJR16	88.47	93.17
NWGL	**89.47**	**93.89**
NWFR	89.33	93.78
KBUS	89.19	93.68

(a) Setting 1: All politicians in the test set are present in the training set.

	All		Absent from train		Present in train	
Model	Accuracy (%)	F1 (%)	Accuracy (%)	F1 (%)	Accuracy (%)	F1 (%)
MAJ	*83.82*	*91.19*	*83.65*	*91.07*	*83.82*	*91.19*
PARTYM	*83.56*	*90.54*	*84.59*	*91.07*	*83.53*	*90.52*
KJR16	84.31	90.73	49.96	62.37	85.51	91.48
NWGL	85.24	91.23	78.40	86.79	85.48	91.38
NWFR	83.65	90.54	79.13	87.66	83.80	90.64
KBUS	**85.98**	**91.96**	**83.36**	**90.52**	**86.07**	**92.01**

(b) Setting 2: An arbitrary 5% of politicians in every session of Congress are absent from the training set.

Table 2: Performance of our proposed models: News augmented-Glove (NWGL), News augmented-Word Frequency (NWFR) and KB Augmented-US (KBUS). These are compared to baselines Majority Class (MAJ) and Majority Party (PARTYM), and to the model from Kraft et al. (2016) (KJR16) for both evaluation settings. Here we report performance averaged across all four sessions of Congress, and highlight the best model excluding MAJ and PARTYM, which are unrealistic solutions.

successfully predict many of the votes of unseen politicians. The KB augmented model, KBUS, outperforms KJR16 by 33.4% points of accuracy, an error reduction of 66.7%.

MAJ and PARTYM outperform the proposed models in Setting 2. However, PARTYM is based on knowledge of the future and so is impractical, and MAJ has no discriminative power for individual politicians.

4.0.3 Comparison of Augmented Models

The KB augmented model KBUS relies on structured contextual information present in relation triples while the news text augmented models NWGL and NWFR rely on unstructured representations of contextual information (average of unigram embeddings). Also, the KB is manually curated while the news augmented models rely on noisier automatically selected and processed sources of information.

In Setting 1 (the setting in which all politicians are present in the training data) all three augmented models perform similarly, with NWGL and KBUS slightly outperforming NWFR (Table 2a). However, in Setting 2 (where some politicians are unseen in the training data) KBUS clearly outperforms NWGL and NWFR, both overall (85.98% accuracy) and specifically on the unseen politicians (83.36% accuracy). We conclude that the structured nature of the additional knowledge provided to this model might allow for more effective use of contextual knowledge. This also suggests that the news text augmented model can be improved by improving the quality of news text features or learning a more clever weighting of unigram embeddings, for example by the CNN embedding approach used in Kornilova et al. (2018).

4.0.4 Comparison with PARTYM

Here we present a deeper analysis of specific politicians to highlight cases where rich contextual information aids prediction as compared to a model only relying on party affiliation.

We examine results for the 108th Congress, where our KB augmented model KBUS achieves an accuracy of 91.39%, while the PARTYM baseline has accuracy of only 85.26%. The congressperson embeddings learned by KBUS also appear to capture party affiliation (as demonstrated by the near-linear separability of parties evident in Figure 1), but they strictly outperform PARTYM for this session. The embeddings learned by

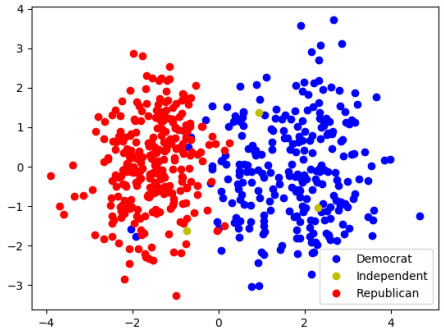


Figure 1: 2-dimensional PCA projected congressperson embeddings by the KBUS model for the 108th congress. Points are colored by party affiliation.

PARTYM primarily err under two circumstances: politicians who do not always vote the party line, and those who identify themselves as Independent. We examine a handful of individual politicians to highlight cases where the proposed models outperform the PARTYM baseline:

Politicians straying party lines : Joe Baca was a congressman from California (D-CA) who was a Democrat during the 108th Congress, but has since changed party twice, due to his Republican-leaning ideology. For Congressman Baca KBUS's accuracy is 90.9%, compared to PARTYM's accuracy of 79.09%. As another example, Jeff Flake (R-AZ) was at that time a congressman recognized as a traditional conservative, but voted with Democrats on issues such as immigration and employment non-discrimination. For Congresssman Flake, KBUS's accuracy is 83.92% while PARTYM's accuracy is 76.78%.

Independent candidates : Joseph Lieberman was an Independent senator from Connecticut (I-CT). For Senator Lieberman, KBUS's accuracy is 91.67% while PARTYM's accuracy is 83.33%. Votes cast by Independents cannot be reliably predicted by considering how other Independents would vote on the same bill.

These results further highlight the importance of a model that is able to combine voting history with rich contextual knowledge in order to accurately predict votes.

5 Related Work

Prior work on predicting roll call votes on bills is primarily focused on using Ideal Point Models, which assume that a politician's ideology and the ideology reflected in a bill lie along a single dimension. In Clinton et al. (2004), an Ideal Point model was trained over both politicians and bills/issues, and at inference time the similarity between politician and bill was used to determine how likely the politician was to vote for the bill. However, most politicians have distinct views on different issues, meaning that the views of one politician cannot be captured in a single dimension. Recent work has taken this into account. For example, Gerrish and Blei (2012) created a model in which each politician's ideal point is adjusted per issue, based on the bill text.

Kraft et al. (2016) used an embedding based model which jointly learns bill and politician embeddings, and can predict how a politician will vote on a bill. As demonstrated in this paper, this model works well when a politician's voting track record has already been established. However, it fails for politicians not in the training data, such as those who have never been elected to office or voted on bills relevant to the issues in the target bill. Although we did not implement any Ideal Point Models, they obviously share this weakness.

There have also been approaches that incorporate additional knowledge about politicians and bills to yield extra insight into politicians' voting patterns. Kornilova et al. (2018) enhanced the embeddings learned by their model by providing bill sponsorship information along with a CNN architecture for learning bill embeddings, while Nguyen et al. (2015) supplemented their model by taking into consideration the type of language legislators use. However, these models share the inability of earlier models to predict voting behavior for new politicians or political candidates. In fact, Kornilova et al. (2018) explicitly note that their model cannot handle unobserved politicians: they say, "During testing, we only include legislators present in the training data". The contributions of Kornilova et al. (2018), among them the CNN-based bill representation and the incorporation of bill metadata, are orthogonal to those in this paper. We leave the exploration of how to combine these two approaches to future work.

Our work builds on the idea of using additional knowledge about politicians to enhance vote pre-

diction performance. The model described by Kraft et al. (2016) serves as our baseline model, and our proposed augmented models build on this baseline by supplementing it with additional knowledge about the politicians. Unlike Kornilova et al. (2018), the additional knowledge we inject is about politicians rather than bills. This allows our proposed models to generalize to politicians unseen at training time.

6 Conclusion and Future Work

In this paper we proposed methods for augmenting a state of the art model (Kraft et al., 2016) for roll call vote prediction with rich sources of additional knowledge to facilitate prediction in cases where the voting record for a politician is unavailable at training time. This is typically the case for new candidates for office or newly elected politicians. We demonstrate that our proposed models outperform a previous state-of-the-art model both when the voting record for all politicians in the test set is known and when the voting record of some politicians is not available at training time.

We propose several avenues for future research. First, researchers could explore richer representations for text, both bill text and news text - for example, CNNs or contextual language models like BERT (Devlin et al., 2019). Second, researchers could explore methods for knowledge augmentation using open information extraction or other ways of automatically constructing KBs, to reduce reliance on manually curated knowledge sources which are subject to bias and quickly become out of date. And third, models could be explored that can take into account interactions between politicians (e.g. changes in majority party, changes in politicians' affiliation, party movement leftward or rightward), between bills (e.g. bill combination or revision), and between politicians and bills across sessions of Congress (e.g. bill revision).

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

Joshua Clinton, Simon Jackman, and Douglas Rivers. 2004. The statistical analysis of roll call data. *American Political Science Review*, 98(2):355–370.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of The Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

John Duckitt and Chris G. Sibley. 2010. Personality, ideology, prejudice, and politics: A dual-process motivational model. *Journal of Personality*, 78(6):1861–1894.

Francis Ferraro, Max Thomas, Matthew Gormley, Travis Wolfe, Craig Harman, and Benjamin. Van Durme. 2014. Concretely annotated corpora. Presented at the NIPS Workshop on Automated Knowledge Base Construction (AKBC).

Sean Gerrish and David M. Blei. 2012. How they vote: Issue-adjusted models of legislative behavior. In *Advances in Neural Information Processing Systems 25*.

Anastassia Kornilova, Daniel Argyle, and Vladimir Eidelman. 2018. Party matters: Enhancing legislative embeddings with author attributes for vote prediction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Peter Kraft, Hirsh Jain, and Alexander M. Rush. 2016. An embedding model for predicting roll-call votes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Robert E Kraut and Steven H Lewis. 1975. Alternate models of family influence on student political ideology. *Journal of Personality and Social Psychology*, 31(5):791.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, and Kristina Miler. 2015. Tea party in the house: A hierarchical ideal point topic model and its application to Republican legislators in the 112th Congress. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Keith T Poole and Howard Rosenthal. 1985. A spatial model for legislative roll call analysis. *American Journal of Political Science*, pages 357–384.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

D. Sunshine Hillygus. 2005. The missing link: Exploring the relationship between higher education and political engagement. *Political Behavior*, 27(1):25–47.

BeamSeg: a Joint Model for Multi-Document Segmentation and Topic Identification

Pedro Mota
Carnegie Mellon University
Pittsburgh, PA, USA
INESC-ID
Instituto Superior Técnico
Lisboa, Portugal
pjdrm@ist.utl.pt

Maxine Eskenazi
Carnegie Mellon University
Pittsburgh, PA, USA
max@cs.cmu.edu

Luísa Coheur
INESC-ID
Instituto Superior Técnico
Lisboa, Portugal
lcoheur@l2f.inesc-id.pt

Abstract

We propose BeamSeg, a joint model for segmentation and topic identification of documents from the same domain. The model assumes that lexical cohesion can be observed across documents, meaning that segments describing the same topic use a similar lexical distribution over the vocabulary. The model implements lexical cohesion in an unsupervised Bayesian setting by drawing from the same language model segments with the same topic. Contrary to previous approaches, we assume that language models are not independent, since the vocabulary changes in consecutive segments are expected to be smooth and not abrupt. We achieve this by using a dynamic Dirichlet prior that takes into account data contributions from other topics. BeamSeg also models segment length properties of documents based on modality (textbooks, slides, *etc.*). The evaluation is carried out in three datasets. In two of them, improvements of up to 4.8% and 7.3% are obtained in the segmentation and topic identifications tasks, indicating that both tasks should be jointly modeled.

1 Introduction

Documents exhibit a content organization that aggregates related text passages in topically coherent segments. Understanding the document structure at the segment level enables efficient content navigation. This has become more relevant with the number of available documents on the Web. The current information landscape allows access to documents describing the same subject, providing alternative views or complementary information. This is advantageous in a variety of scenarios. For example, students have at their disposal several learning materials and might need to find a particular topic segment that best suits their learning needs. Finding such documents is an easy task since search engines are capable of returning documents conveying related information. However, if search engines are effective in retrieving these documents, the task of putting them into a coherent picture remains a challenge (Shahaf et al., 2012). Automatically finding document segments – text segmentation – and identifying which ones discuss the same topic – topic identification – addresses this issue (Jeong and Titov, 2010).

Text segmentation and topic identification have been used as intermediate steps in a variety of natural language processing tasks, including summarization (Radev et al., 2004), opinion mining (Murakami et al., 2009), semantic and information retrieval (Purver, 2011; Amoualian et al., 2017). The improvements they brought spurred research in text segmentation. Invariably, all works resort to the lexical cohesion theory (Halliday and Hasan, 1976), which postulates that discourse structure is correlated to the use of cohesive vocabulary. Thus, segments can be identified by detecting vocabulary changes. Most approaches either consider segmentation and identification separately and/or do not take into account all documents in the dataset (single-document approach). Recently, some works studied these phenomena in collections of related documents (Jeong and Titov, 2010; Mota et al., 2016). These multi-document models assume that documents describing the same topic have similar lexical cohesion properties; an example of this phenomenon with similar segments but in different documents is depicted in Figure 1. Thus, better likelihood estimations can be obtained if all documents are taken into account (Mota et al., 2016). In this work, we expand the multi-document lexical cohesion idea by hypothesizing that vocabulary relationships between different segments exist. For example, if a word is heavily used in one segment, it is likely that it continues to appear in the following one, though less frequently. Modeling such interactions can lever-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 582–592
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

age topic segmentation algorithms. We also explore the role of modality in the multi-document scenario. Previous approaches treat all documents equally, but it is plausible that we can improve segmentation by making assumptions about the expected segment length on a document modality basis. For example, segments in slide presentations are expected to be shorter than in video lectures.

We propose BeamSeg, a Bayesian unsupervised topic modeling approach to breaking documents in coherent segments while identifying similar topics. The generative process assumes that segments can share the same topic and, consequently, are generated from the same lexical distribution. Lexical cohesion is achieved by having higher segmentation likelihoods when the probability mass is concentrated in a narrow subset of words. This is in the same spirit as topic modeling approaches such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003), but here the inherent topics are constrained to the linear discourse structure. To model interactions between lexical distributions, we use a dynamic prior, which assumes that the word probabilities change smoothly across topics. To model segment length characteristics, we assign prior variables conditioned on document modality.

The linear segmentation constraint has been used to make inference tractable by exhaustively exploring the segmentation space to obtain the exact maximum-likelihood estimation (Eisenstein and Barzilay, 2008). Given a multi-document setting, this is not feasible, as segments can share topics. We address this issue using a beam search algorithm, which allows the inference procedure to recover from early mistakes. In our experiments, we show that BeamSeg is able to perform well when segmenting learning materials, where previously single-document models obtained better results (Mota et al., 2018). We also observe that topic identification is more accurately determined in a joint model, as opposed to a pipeline approach (performing the tasks sequentially), indicating that both problems should be modeled simultaneously.

We summarize our contributions as follows:

- A novel joint model for topic segmentation and identification with a dynamic prior.
- An inference procedure based on a beam search algorithm.
- A study on how different modality-based segment length priors influence segmentation.

The source code is available in the following repository: `github.com/pjdrm/BeamSeg`.

2 Related Work

Following the lexical cohesion theory, segmentation algorithms identify spans of text with prominent vocabulary changes. The main difference between algorithms is how lexical cohesion is implemented: some resort to lexical similarity; the remaining follow a probabilistic approach.

Lexical approaches rely on a similarity metric between sentences, usually the cosine. A classic method is TextTiling (Hearst, 1997), which assumes that topic boundaries are found in consecutive sentences with a low similarity value; several other works built on this idea (Galley et al., 2003; Balagopalan et al., 2012). C99 (Choi, 2000) is another lexical approach, and uses a similarity matrix in a divisive clustering to obtain segments. MinCut (Malioutov and Barzilay, 2006) casts segmentation in a minimum cut graph partitioning problem. The graph has a node for each sentence; edges are weighted using lexical similarity. Long-distance textual relationships are modeled by connecting all sentences. Affinity Propagation Segmentation (Kazantseva and Szpakowicz, 2011) also models such relationships but uses affinity propagation clustering (Frey and Dueck, 2007). The algorithm creates a factor graph and maximizes the segment similarity sum function. Alemi and Ginsparg (2015) proposed the Content Vector Segmentation (CVS) sentence vector representation based on segment word embeddings. Using this representation in C99 improves bag-of-words results.

In another line of research, Wang et al. (2017) combined learning to rank and a convolutional neural network to learn a coherence function between text pairs; higher-ranked pairs are likely to be segments. Despite a promising approach, state-of-the-art results were not achieved. Also following an approach using neural networks, is the SECTOR algorithm (Arnold et al., 2019), which uses a topic embedding trained based on utterance topic classification. Following the network architecture from (Koshorek et al., 2018), two stacked LSTM layers are used to decode word embedding representation of utterances. To recover segmentation, a TextTiling approach is applied to the topic embedding layer. The evaluation results show that SECTOR is able to improve a C99 baseline.

Just as we introduced average **velocity** we will now describe **average acceleration**. Notice when **velocity** changes ... over **time**. And ... introduce an **average acceleration** ... The **average acceleration** between **time** t2 ... And the dimension ... secs per **time** squared.	**Acceleration** We say ... changing **velocity** are "accelerating" **Acceleration** is the "Rate of change of **velocity**" You hit the accelerator to speed up ... it's true you also hit ... friction is slowing ... **Average acceleration** Unit of **acceleration**: (m/s)/s=m/s2	The **acceleration** of a particle ... rate of change of **velocity** ... **time**. Average acceleration ... is v2 - v1 t2 - t1 ... **Acceleration** may be positive, negative or zero. Zero **acceleration** means we have constant **velocity**. Note that the direction and **acceleration** need not coincide.

Figure 1: Examples of segment excerpts from video, slide presentation, and PDF documents describing the acceleration topic. Words in bold depict shared vocabulary across segments.

Probabilistic approaches to segmentation follow a setup similar to the LDA model: words are assigned to topics such that probability mass is distributed on a small set of topically relevant words. In order to adapt this idea to segmentation, the model needs to be able to determine if sentences belong to the same topic (or mixture of topics). An example of such adaptation is the single-document segmentation model PLDA (Purver et al., 2006), where topic proportions are shared by sentences within the same segment. Segmentation is then determined through a binary topic shift sentence variable. Models such as TopicTiling (Riedl and Biemann, 2012), Structured Topic Model (STM) (Du et al., 2013), and NTSeg (Jameel and Lam, 2013) extend this LDA-based approach to segmentation. In all these approaches, topic identification is not possible since all segments are a mixture of topics.

In this paper, we adopt a probabilistic multi-document view on segmentation. Only two other models follow this approach: MultiSeg (Jeong and Titov, 2010) and Bayesseg-MD (Mota et al., 2016). MultiSeg uses a two-level LDA model where documents are generated using local and global topics. Local topics are specific to a document; global topics are shared between documents. Documents are mixtures of topics, but each segment is generated by a single topic, lending itself to a joint model of segmentation and topic identification. The multi-document aspect of the model stems from topic proportions being inferred from the whole dataset. In the experiments, this joint modeling outperforms a pipeline strategy that performs these tasks sequentially.

The other multi-document model, Bayesseg-MD, is an extension of Bayesseg (Eisenstein and Barzilay, 2008). In Bayesseg, sentences from the same segment are assigned the same topic. The inference procedure affords an exact maximum-likelihood estimation by exploring the segmentation space with a dynamic programming algorithm. This approach cannot be applied to multi-document segmentation since the hidden topic variables are integrated out; other single-document models following this approach also have this problem (Eisenstein, 2009; Malmasi et al., 2017). Bayesseg-MD sidesteps this problem by using lexically similar sentences from other documents. The word counts of such sentences are added to the segment likelihood estimation to reduce data sparseness. Despite using all documents for segment likelihood estimations, topic identification is not available. In this paper, we address these issues by designing an inference algorithm that explicitly tracks segment topic assignments.

3 BeamSeg Model

We implement lexical cohesion in a Bayesian setting in a generative process where segments with the same topic are drawn from the same multinomial language model. Thus, all u utterances with a topic k have their bag-of-words representation $\mathbf{x}_u$ drawn from language model ϕ_{z_u}; z_u is the hidden topic variable of u. We constrain the model to yield linear segmentations by having topics occurring at most once per document. This induces higher likelihood segmentations to have language models concentrating probability mass on a small subset of the vocabulary. Conversely, low likelihood segmentations spread the probability mass on a broad set of words. This modeling behavior is attuned to the lexical cohesion theory. Multi-document segmentation emerges by assuming that topics are shared across documents.

During inference, we want to find the hidden

set of language models Φ and the topic vector assignment $\mathbf{z}$ that maximize the likelihood of the joint distribution of the model. Since we only care about segmentation, this process can be simplified by analytically marginalizing out the hidden language models Φ. This enables search to be carried out only in the segmentation space. Therefore, inference amounts to finding the segmentation $\hat{\mathbf{z}} = \text{argmax}_{\mathbf{z}}\, p(\mathbf{X}|\mathbf{z})p(\mathbf{z})$. Using the marginalized joint likelihood, an approximation of $\hat{\mathbf{z}}$ is obtained using a beam search algorithm.

3.1 Language Models

Using the previous setup, we define the joint likelihood as follows:

$$p(\mathbf{X}|\mathbf{z}, \Phi) = \prod_{k}^{K} p(\phi_k|\beta) \prod_{u}^{U} p(\mathbf{x}_u|\phi_{z_u}), \quad (1)$$

where $\mathbf{X}$ is the set of all U utterances in the dataset; K is the number language models; and β are the Dirichlet prior parameters from which Φ is drawn.

The marginalization process is performed by appealing to the conjugacy between multinomial language models and the Dirichlet prior. This allows the conjugate Dirichlet distribution to integrate to one, leaving the marginalized joint likelihood expression with the normalizing constants:

$$\begin{aligned} p(\mathbf{X}|\mathbf{z}) &= \int p(\mathbf{X}|\mathbf{z}, \Phi)p(\Phi|\beta)d\Phi \quad (2) \\ &= \Big(\frac{\Gamma(W\beta)}{\Gamma(\beta)^W}\Big)^K \prod_{k=1}^{K} \frac{\prod_{w=1}^{W} \Gamma(n_{U,w}^k + \beta)}{\Gamma(n_U^k + \beta)}, \end{aligned}$$

where W is the vocabulary set; $n_{U,w}^k$ is number of times word w is assigned topic k in all U utterances of the document collection; n_U^k is number of times topic k appears in U; and the symbol Γ refers to the Gamma function. The resulting expression in Equation 2 corresponds to the product of the individual topic likelihoods, comprised of segments from different documents.

3.2 Segment Length Prior

The $\hat{\mathbf{z}} = \text{argmax}_{\mathbf{z}}\, p(\mathbf{X}|\mathbf{z})p(\mathbf{z})$ expression we want to maximize to obtain the most likely segmentation puts a prior, $p(\mathbf{z})$, on the segment length of documents. Given the approach of searching the segmentation space only during inference, we do not require the mathematical conveniences of conjugacy for the segment length prior. In this perspective, we can plug in different distributions to see how they behave during the segmentation task. One of such distribution is the Beta-Bernoulli, which has been used before in a probabilistic approach to segmentation (Purver et al., 2006):

$$p(\mathbf{z}) = \left(\frac{\Gamma(2\gamma)}{\Gamma(\gamma)^2}\right)^D \prod_{d=1}^{D} \frac{\Gamma(n_1^d + \gamma)\Gamma(n_0^d + \gamma)}{\Gamma(U_d + 2\gamma)}, \quad (3)$$

where D is the number of document in the dataset, U_d is the total number of utterances in document d, n_1^d is the number of segments in d, n_0^d the number of non-segment boundary utterances in d, and γ the hyperparameter of the Beta distribution.

We also propose a Gamma-Poisson distributed segment length prior. In this setup, we assume that the document topic shift probabilities are drawn from a Gamma prior parameterized by α and β:

$$p(\mathbf{z}) = \left(\frac{\beta^\alpha}{\Gamma(\alpha)}\right)^D \prod_{d=1}^{D} \frac{\Gamma(n_1^d + \alpha)}{(U_d + \beta)^{n_1^d + \alpha}} \quad (4)$$

Applying priors based on document modality can be done by assuming they are known *a priori*, which is the approach we take. It is only necessary to have dedicated hyperparameters for each modality and apply them accordingly when computing segmentation likelihood. This means we are encoding in the model our prior beliefs about the segment length of each modality. Nonetheless, if the lexical cohesion in a hypothesized segment is strong enough, the model will identify it even if the length is not inline with the prior.

3.3 Dynamic Language Model Prior

The previous priors assume that language model's draws are independent of each other, and, thus cannot encode relationships between them. This is not a reasonable assumption in datasets with documents following an overarching subject. We hypothesize that in these cases, language models change smoothly across topics by establishing a dynamic between the previous and the current prior parameters. This time series modeling of topics can be found in other works (Blei and Lafferty, 2006b,a; Du et al., 2013; Jahnichen et al., 2018). In BeamSeg, we adopt a similar perspective to topic tracking (Watanabe et al., 2011) for modeling such interactions. We factor the β in $\alpha_k \hat{\phi}_{k'}$, a precision and mean language model word

probabilities parameters. Assuming some ordering between the topics, k indexes the topic parameters, and k' the parameters of the previous topic. The α_k precision represents the persistence of word usage throughout topics; $\hat{\phi}_k$ models the language model dynamics by assuming that the mean word probabilities at k are the same as those at k'. This entails that there is a single chain of language models, which contrasts with the multiple chains in the original topic tracking model.

To compute the likelihood of the joint under this prior, it is necessary to determine the parameters for all $k \in K$. This is a two-fold process, where we first update the α_k precision parameter using the expression derived from Minka (2000):

$$\alpha_k = \alpha_k \frac{\sum_{w}^{W} \hat{\phi}_{k'w}(\Psi(n_{kw} + \alpha_k \hat{\phi}_{k'w}) - \Psi(\alpha_k \hat{\phi}_{k'w}))}{\Psi(n_k + \alpha_k) - \Psi(\alpha_k)}, \tag{5}$$

where n_{kw} is the number of times word w appear in k; n_k is the total number of words in k; and Ψ is the digamma function. Then, we update the mean word probability parameters:

$$\hat{\phi}_{kw} = \frac{n_{kw} + \alpha_k \hat{\phi}_{k'w}}{n_k + \alpha_k} \tag{6}$$

The update equations are sequentially applied according to a fixed topic order. By following this process, we model long-range dependencies by taking into account the data contribution at each k. Finally, we plug-in the obtained prior parameters in the join likelihood formula in Equation 2.

3.4 Beam Search for Inference

Following Bayesseg (Eisenstein and Barzilay, 2008), inference is viewed as an optimization problem, where the target segmentation maximizes the objective function defined by the joint likelihood. Contrary to Bayesseg, we assume that language models aggregate segments from different documents, making an exhaustive exploration of the segmentation space intractable. To address this problem we combine beam search and a greedy segmentation procedure. Other considered inference alternatives include Gibbs sampling (Bishop, 2006) and Variational Inference (Ghahramani et al., 2008). The difficulty in applying Gibbs sampling is its slow convergence to the stationary distribution, due to the tight coupling of the variables induced by the linear segmentation constraint. A similar problem occurs in the variational inference procedure from Eisenstein (2009), where variational parameters and segmentation are iteratively estimated.

We define $\mathbf{z}_j^*$ as the segmentation that maximizes the objective function up to utterance j. Considering the topic assignment $z_j = k$ and the previous segmentation $\mathbf{z}_{j-1}$, the value for the objective function is written,

$$s(k, j, \mathbf{z}_{j-1}) = p(\{\mathbf{x}_0 ... \mathbf{x}_j\} | \mathbf{z}_{j-1}, z_j = k) \tag{7}$$

Using a recursive definition, we obtain the optimal segmentation using:

$$\mathbf{z}_j^* = \underset{k \in K}{\operatorname{argmax}}\, s(k, j, \mathbf{z}_{j-1}^*) \tag{8}$$

This is a greedy approach since it makes incremental decisions to find the highest likelihood segmentation. This is an error-prone procedure since we should take into account subsequent utterances to discover higher likelihood segmentations. Moreover, once a mistake is made, we cannot recover from it. To address this problem, we add a beam search feature to the algorithm. This is achieved by keeping track of all topic assignments, instead of just the highest likelihood one. At the end of each recursive step, we prune the top-n segmentations.

4 Experiments

We now describe the experimental setup and report the results for the target tasks.

4.1 Datasets

Currently, there are two multi-document segmentation datasets with different document modalities. One of the datasets is comprised of learning materials describing the subject of Adelson-Velsky and Landis' (AVL) trees (Mota et al., 2016). The available modalities are video transcripts, PPT, and HTML. In total, the dataset contains 10 documents, 85 segments, and 17 topics. The other dataset also contains learning materials but from the Physics domain (Mota et al., 2018). In addition, this dataset also has PDF modality. The dataset has 141 documents, 739 segments, and 135 topics from 7 different Physics subjects. This dataset does not provide topic identification labels

for the segments. Therefore, we manually annotated it with this information. In this context, we made an inter-annotator agreement study for the 'Introduction to Kinematics' subject with two annotators. A 0.69 Fleiss-kappa (Shrout and Fleiss, 1979) agreement value was obtained, showing that annotators had a similar perception of whether segments share the same topic. Most of the disagreement cases are due to considering textual and plot-based explanations as different topics.

In addition to the previous datasets, we also used Biography documents from Jeong and Titov (2010). The dataset contains 116 documents regarding 29 personalities; 4 documents per personality with a total of 240 segments; the number of topics is 405; all documents have the same HTML modality. The Biography domain has different topic development characteristics from the previous domains. The documents have fewer and shorter segments when compared with the AVL and Physics domains, leaving less room for topics to be described. All datasets were preprocessed by stemming and stop words were removed.

4.2 Segmentation Experiments

In the experiments, we benchmark syntax similarity and probabilistic approaches: C99, CVS, Bayesseg, PLDA, Bayesseg-MD, and MultiSeg. The hyperparameter tuning of the models is done on a development set. In the Biography dataset, we use documents from one of the personalities. For MultiSeg we use the configurations provided by the authors. In the Physics domain, we use ten documents from one of the subjects. The obtained tuning is also used for the AVL trees domain since both datasets have pedagogical content. The Gibbs sampling for PLDA run for 20000 iterations with a burn-in period of 1000 and a lag value of 200. In BeamSeg, we investigate the role of two factors in segmentation: using the dynamic *vs.* an independent language model prior, and using a modality-based segment duration prior *vs.* using a single prior variable. The beam size was set to 200.

To measure performance, we use the standard Window Difference (WD) metric (Pevzner and Hearst, 2002). WD slides a window through a document and penalizes segmentations according to the difference between the number of expected segment boundaries and the predicted ones. This gives partial credit to near-miss situations. The metric is calculated as follows:

$$\text{WD} = \frac{1}{N-k} \sum_{i=1}^{N-k} |ref - hyp| \neq 0, \quad (9)$$

where N is the length of the document and k the window size. WD is a penalty score between 0 (the best value) and 1. For consistency, we take the output segmentations from all systems and evaluate it using the same software (the python module `segeval` (Fournier, 2013)).

The WD average results for the baseline are in Table 1. In the Biography dataset, MultiSeg is the best performing model, improving the WD of Bayesseg-MD by 0.05. In the AVL dataset, the best results are obtained by Bayesseg-MD. The difference to the second best result, Bayesseg, is 0.02. For the Physics dataset, the single-document model Bayesseg achieves the best results with a WD difference of 0.01. These results show that the performance of the algorithms varies across the different datasets. This suggests that the different modeling approaches do not generalize well to the different characteristics of the datasets. The Biography dataset is characterized by short segments, which does not leave much room for lexical cohesion to be observed. This contrasts with the AVL and Physics datasets where the segments are longer and describe an overarching topic.

Table 1: Segmentation baseline average WD results.

	Bio	**AVL**	**Physics**
C99	0.61	0.59	0.54
PLDA	0.58	0.55	0.49
CVS	0.54	0.45	0.43
Bayesseg	0.53	0.39	**0.42**
Bayesseg-MD	0.42	**0.37**	0.43
MultiSeg	**0.37**	0.41	0.44

The results using different prior configurations are in Table 2. In the table, the LMP and SLP columns correspond to the language model and segment length priors. In the Biography dataset, we can see that using the dynamic LMP instead of the independent improves the the Beta-Bernoulli and Gamma-Poisson results by 0.01 and 0.09, respectively. In the AVL dataset, the dynamic LMP improves the best WD results of the independent LMP by 0.02. When comparing the scope results of the dynamic LMP in the AVL dataset, we observe further improvements when

Table 2: BeamSeg average WD results. The SLP column depicts the Beta-Bernoulli (BB), and Gamma-Poisson (GP) distributions. The scope indicates if the SLP is modality-based (M) or if there is one variable for the whole dataset (D). The Biography dataset has one modality, and, thus, only the D scope exists.

LMP	SLP	Scope	Bio	AVL	Physics
Ind	BB	D	0.54	0.39	0.45
		M	–	0.40	0.42
	GP	D	0.58	0.40	**0.40**
		M	–	0.43	0.42
Dyn	BB	D	0.53	0.44	0.54
		M	–	0.38	0.42
	GP	D	0.49	0.38	0.47
		M	–	**0.37**	**0.40**

Table 3: Number of exact segment boundary matches between hypothesis and reference segmentations.

LMP	SLP	Scope	Bio	AVL	Physics
Ind	BB	D	88	1	16
		M	–	1	8
	GP	D	15	1	5
		M	–	1	20
Dyn	BB	D	147	3	34
		M	–	4	39
	GP	D	244	2	19
		M	–	5	46

using the modality-based SLP; the results differences are 0.06 and 0.01, respectively. In the Physics dataset, a dynamic LMP combined with the modality-based Gamma-Poisson SLP obtains the best results tied with the independent LMP and dataset-based Gamma-Poisson SLP. It should be noted that the former configuration better generalizes across the different datasets since it obtains better results in the Biography and AVL datasets; the WD differences are 0.09 and 0.03, respectively. Looking at the scope results of the dynamic LMP, we observe that the Beta-Bernoulli and the Gamma-Poisson perform better when using the modality prior (0.12 and 0.07 improvements).

WD is a metric that assesses the overall quality of a segmentation, accounting for different types of errors. This can make the WD scores of two very different segmentations to be close, which is the case of the previous results. For example, a segmentation that has no segments and another that only has misplaced segments will have similar WD scores despite being different. To show that the different prior configurations output significantly different segmentations, we provide the counts of the exact segment boundary matches in Table 3. From these results, we can observe that using a dynamic LMP can increase the number of boundary up to 229. A similar observation can be made when comparing the dataset and modality scopes, where the increases are up 27 segments. These increases in exact boundary matches show that despite the small differences in WD the impact on how the segmentation looks like is significant. Therefore, we conclude that using a dynamic LMP with a modality Gamma-Poisson SLP is necessary to achieve the best results.

Comparing BeamSeg's results to the baseline, we see that in the Biography dataset MultiSeg performs better by a 0.12 margin. The main difference between the segmentation of the two approaches is that BeamSeg outputs fewer segments, which is a disadvantage since this dataset has a high number of short segments. In the AVL dataset, the performance is similar to Bayesseg-MD. Looking at the individual documents shows that BeamSeg has better results in five out of ten documents, one tie, and two documents where the WD difference is 0.01. This leaves Bayesseg-MD to perform significantly better only in two documents. Therefore, BeamSeg is more consistent in this dataset. In the Physics dataset, BeamSeg improves the Bayesseg baseline by 4.8%. Taking into account the result analysis, we conclude that BeamSeg's performance depends on the characteristics of the datasets. In datasets where topic development is prominent across the segments (AVL and Physics), BeamSeg is the model with the most consistent results. This is only possible when using a dynamic LMP and a modality Gamma-Poisson SLP, showing that both modeling aspects are relevant to obtain the best segmentation.

To understand the behavior of the priors we provide a segmentation example in Figure 2. From the example, we see that the main difference between the independent and dynamic LMPs is the number of segments. In the independent LMP, the number of segments is low, especially when using the dataset SLP. For the modality SLP, the number of segments is higher but they tend to be mis-

placed. When using dynamic LMP, the behavior changes at the SLP level. The dataset SLP outputs more segments than the modality version. However, most segments do not match the reference. The modality SLP finds fewer segments, but they tend to be more accurate. This makes sense since the over-segmentation of the dataset SLP might be related to the bias towards documents with short segments, and the modality prior is able to adjust to a wider variety of documents.

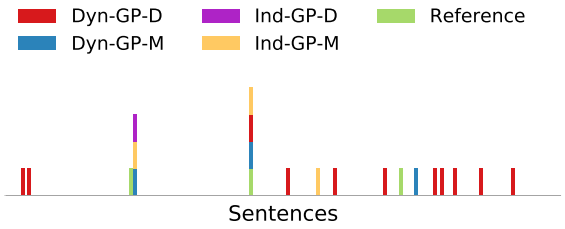


Figure 2: Physics document segmentation using different prior configurations in BeamSeg. Bars with the same color represent segments of the same prior configuration. The names of the configurations start with the LMP type, followed by the SLP, and its scope.

4.3 Topic Identification Experiments

We use the previous datasets to evaluate topic identification and compare multi-document joint models to a pipeline approach. In the pipeline approach, we evaluate clustering and graph-community detection algorithms. The clustering algorithms take the golden standard segments and identify segments sharing the same topic if they are assigned the same cluster. Several clustering algorithms are surveyed (Aggarwal and Reddy, 2014): DBSCAN, Mean Shift, and NMF. For the graph-community detection approach, word communities are obtained from the segments. Then, based on lexical similarity, segments are assigned to one of the communities (Mota et al., 2018). If two segments are assigned to the same community, they share the same topic. Several graph-community detection algorithms are surveyed (Fortunato, 2010): Bigclam, Label Propagation, CNM, Walktraps, and Leading Eigenvector. For conciseness, we only report the results of the best algorithms.

To measure the performance, we use the standard B^3 clustering metric (Amigó et al., 2009). B^3 decomposes uses item-wise precision and recall. Precision represents how many items in the same cluster belong to its class. Recall represents how many items from a class appear in the cluster. The final B^3 value combines precision and recall:

$$B^3 = \frac{1}{0.5(\frac{1}{Pre}) + 0.5(\frac{1}{Rec})} \quad (10)$$

The baseline results are depicted in Table 4. In this benchmark, the pipeline approach performs better than the joint model in all datasets. The differences range between 0.04 and 0.14 in B^3 score. The DBSCAN clustering approach obtains the best performance in the Biography dataset by a 0.09 margin.The Louvain graph-community detection approach obtains the best results in the AVL and Physics datasets with result differences to DBSCAN of 0.04 in both cases.

Table 4: Topic identification baseline results.

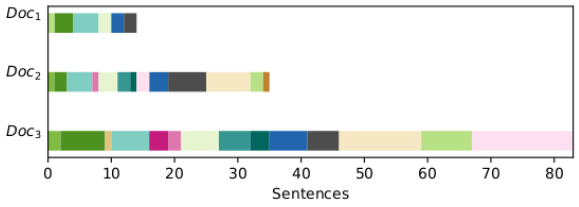
	Bio	AVL	Physics
DBSCAN	**0.66**	0.33	0.34
Louvain	0.57	**0.37**	**0.38**
MultiSeg	0.52	0.29	0.30

Table 5 shows the results for different prior configurations. In the Biography domain, we observe that the dynamic LMP improves the results of both SLPs; 0.03 and 0.16, for the Beta-Bernoulli and Gamma-Poisson, respectively. In the AVL datasets, three different configurations obtain the best performance. In the Physics dataset, the dynamic LMP modality Gamma-Poisson SLP performs better. In this case, using a modality SLP instead of the dataset affords a 0.11 improvement. Comparing the independent and dynamic LMPs, we see that the former improves the results by 0.05. This shows that both modeling aspects are contributing for the best results.

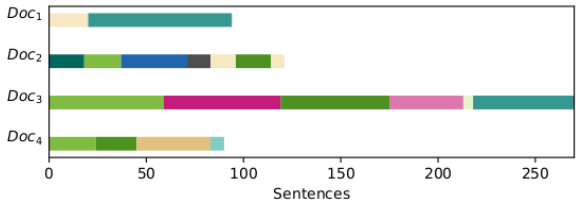
Table 5: BeamSeg topic identification results.

LMP	SLP	Scope	Bio	AVL	Physics
Ind	BB	D	0.51	0.35	0.36
		M	–	**0.39**	0.38
	GP	D	0.37	0.38	0.35
		M	–	0.36	0.37
Dyn	BB	D	0.54	**0.39**	0.30
		M	–	0.32	0.34
	GP	D	0.53	0.38	0.31
		M	–	**0.39**	**0.41**

Comparing BeamSeg's best results to the baseline, we observe that it is only outperformed by DBSCAN in the Biography dataset (a 19.7% difference). DBSCAN obtains better results by putting segments it cannot group in individual clusters, which keep the larger clusters clean. In BeamSeg, the number of identified topics (clusters) is lower, a 385 difference to DBSCAN, which ends up forcing wrong topic segment assignments. In the AVL dataset, BeamSeg improves the Louvain baseline by 5.1%. The topic identification behavior of both approaches is different from the Biography dataset. Louvain only outputs 7 clusters whereas the reference has 17 topics. This is related to the topic development aspect across segments, which makes them hard to distinguish. BeamSeg obtains a higher B^3 score because it is able to identify 15 topics, a number closer to the reference, and, consequently, assign topics more appropriately. In the Physics dataset, BeamSeg improves the baseline by 7.3%. The topic identification patterns are similar to the ones observed in the AVL dataset with BeamSeg outputting more topics than Louvain, 70 and 48 topics, respectively. Another observation is that the performance differences between the Biography and the other datasets are related to the topic structure complexity. In the Biography dataset, there is a tendency for the topic order to persist across documents, whereas in the other datasets the interweaving of the topics is not as regular. This is depicted in Figure 3, where color changes represent a topic changes and similar topics have the same color. In Figure 3a (Biography domain) we can see that the colors sequences in different documents are similar whereas in Figure 3 (Physics domain) the sequence is not constant. Connecting the topic structure differences with the topic order assumptions in BeamSeg explains the performance differences.



(a) Documents from the Biography domain.



(b) Documents from the Physics domain.

Figure 3: Topic identification examples.

5 Conclusions and Future Work

In this work, we propose BeamSeg, an unsupervised Bayesian algorithm that jointly segments documents and identifies topical relationships using a beam search procedure to find high likelihood segmentations during inference. Relationships between topics are modeled using a dynamic prior encoding that word distributions change smoothly in documents with an overarching subject. BeamSeg also models segment length properties based on document modality. To evaluate segmentation, single and multi-document algorithms were used as a baseline. For topic identification, we compared BeamSeg to MultiSeg, another joint model, as well as a pipeline approach. In both tasks, BeamSeg obtains the best results in two of the datasets used for evaluation. The conclusion from the evaluation is that BeamSeg is effective in datasets with prevalent topic development throughout document segments. To achieve the best performance, it is necessary to use a combination of a dynamic LMP with a modality Gamma-Poisson SLP. Therefore, the proposed modeling assumptions fit the data well. This supports the hypothesis that lexical cohesion is a cross-document phenomenon and can be used to leverage multi-document segmentation and topic identification.

Regarding future work, one of the concerns is that the proposed inference procedure is a maximum likelihood estimation approach. Ideally, we want to access the full posterior distribution since it finds more accurate parameters. Another concern is the raw assumption that there is a shared topic ordering among all documents. We believe that addressing these issues will allow BeamSeg to improve its results and to consistently perform in datasets with different characteristics.

Acknowledgements

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019, and through the Carnegie Mellon Portugal Program under Grant SFRH/BD/51917/2012.

References

Charu C. Aggarwal and Chandan K. Reddy, editors. 2014. *Data Clustering: Algorithms and Applications*. CRC Press.

Alexander A Alemi and Paul Ginsparg. 2015. Text segmentation based on semantic word embeddings. *arXiv e-prints*, page arXiv:1503.05543.

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. 2009. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):461–486.

Hesam Amoualian, Wei Lu, Éric Gaussier, Georgios Balikas, Massih-Reza Amini, and Marianne Clausel. 2017. Topical coherence in lda-based models through induced segmentation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1799–1809.

Sebastian Arnold, Rudolf Schneider, Philippe Cudr-Mauroux, Felix A. Gers, and Alexander Lser. 2019. Sector: A neural model for coherent topic segmentation and classification. *Transactions of the Association for Computational Linguistics*, 7:169–184.

Arun Balagopalan, Lalitha L. Balasubramanian, Vidhya Balasubramanian, Nithin Chandrasekharan, and Aswin Damodar. 2012. Automatic keyphrase extraction and segmentation of video lectures. In *Proceedings of the International Conference on Transformations in Engineering Education*, pages 152–162.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer-Verlag, New York.

D. Blei and J. Lafferty. 2006a. Correlated Topic Models. In *Advances in Neural Information Processing Systems*, volume 18, pages 147–154. MIT Press.

David M. Blei and John D. Lafferty. 2006b. Dynamic topic models. In *Proceedings of the International Conference on Machine Learning*, pages 113–120.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Freddy Y. Y. Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics Conference*, pages 26–33.

Lan Du, Wray L. Buntine, and Mark Johnson. 2013. Topic segmentation with a structured topic model. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200.

Jacob Eisenstein. 2009. Hierarchical text segmentation from multi-scale lexical cohesion. In *Proceedings of North American Chapter of the Association for Computational Linguistics*, pages 353–361.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 334–343.

Santo Fortunato. 2010. *Physics Reports*, 486(3-5):75 – 174.

Chris Fournier. 2013. Evaluating text segmentation using boundary edit distance. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1702–1712.

Brendan J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science*, 315(5814):972–976.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the Annual Meeting on Association for Computational Linguistics*, pages 562–569.

Z. Ghahramani, J. Sung, and S. Bang. 2008. Latent-space variational bayes. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 30:2236–2242.

Michael A.K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman, London.

Marti A. Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Patrick Jahnichen, Florian Wenzel, Marius Kloft, and Stephan Mandt. 2018. Scalable generalized dynamic topic models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, pages 1427–1435.

Shoaib Jameel and Wai Lam. 2013. An unsupervised topic segmentation model incorporating word order. In *Proceedings of the International Conference on Research and Development in Information Retrieval*, pages 203–212.

Minwoo Jeong and Ivan Titov. 2010. Multi-document topic segmentation. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 1119–1128.

Anna Kazantseva and Stan Szpakowicz. 2011. Linear text segmentation using affinity propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 284–293.

Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 2018. Text segmentation as a supervised learning task. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 2 (Short Papers), pages 469–473.

Igor Malioutov and Regina Barzilay. 2006. Minimum cut model for spoken lecture segmentation. In *Proceedings of the International Conference on Computational Linguistics and the Annual Meeting of the Association for Computational Linguistics*, pages 25–32.

Shervin Malmasi, Mark Dras, Mark Johnson, Lan Du, and Magdalena Wolska. 2017. Unsupervised text segmentation based on native language characteristics. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1457–1469. Association for Computational Linguistics.

Thomas P. Minka. 2000. Estimating a dirichlet distribution. Technical report.

Pedro Mota, Luísa Coheur, and Maxine Eskénazi. 2018. Efficient navigation in learning materials: An empirical study on the linking process. In *Artificial Intelligence in Education*, pages 230–235.

Pedro Mota, Maxine Eskenazi, and Luísa Coheur. 2016. Multi-document topic segmentation using bayesian estimation. In *Proceedings of the International Workshop on Semantic Multimedia*, pages 443–447.

Koji Murakami, Shouko Masuda, Suguru Matsuyoshi, Eric Nichols, Kentaro Inui, and Yuji Matsumoto. 2009. Annotating semantic relations combining facts and opinions. In *Proceedings of the Third Linguistic Annotation Workshop (LAW III)*, pages 150–153. Association for Computational Linguistics.

Lev Pevzner and Marti A. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Matthew Purver. 2011. Topic segmentation. In *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, pages 291–317.

Matthew Purver, Thomas L. Griffiths, Konrad P. Körding, and Joshua B. Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *Proceedings of the International Conference on Computational Linguistics and the Annual Meeting of the Association for Computational Linguistics*, pages 17–24.

Dragomir R. Radev, Hongyan Jing, Malgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Inf. Process. Manage.*, 40(6):919–938.

Martin Riedl and Chris Biemann. 2012. Topictiling: A text segmentation algorithm based on lda. In *Proceedings of Association for Computational Linguistics Student Research Workshop*, pages 37–42.

Dafna Shahaf, Carlos Guestrin, and Eric Horvitz. 2012. Trains of thought: Generating information maps. In *Proceedings of the International Conference on World Wide Web*.

Patrick E. Shrout and Joseph L. Fleiss. 1979. Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin*.

Liang Wang, Sujian Li, Yajuan Lv, and Houfeng WANG. 2017. Learning to rank semantic coherence for topic segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1340–1344.

Shinji Watanabe, Tomoharu Iwata, Takaaki Hori, Atsushi Sako, and Yasuo Ariki. 2011. Topic tracking language model for speech recognition. *Computer Speech Language*, 25(2):440–461.

MrMep: Joint Extraction of Multiple Relations and Multiple Entity Pairs Based on Triplet Attention

Jiayu Chen, Caixia Yuan, Xiaojie Wang and **Ziwei Bai**
Center of Intelligence Science and Technology
School of Computer Science
Beijing University of Posts and Telecommunications
{jiayuchen, yuancx, xjwang, bestbzw}@bupt.edu.cn

Abstract

This paper focuses on how to extract multiple relational facts from unstructured text. Neural encoder-decoder models have provided a viable new approach for jointly extracting relations and entity pairs. However, these models either fail to deal with entity overlapping among relational facts, or neglect to produce the whole entity pairs. In this work, we propose a novel architecture that augments the encoder and decoder in two elegant ways. First, we apply a binary CNN classifier for each relation, which identifies all possible relations maintained in the text, while retaining the target relation representation to aid entity pair recognition. Second, we perform a multi-head attention over the text and a triplet attention with the target relation interacting with every token of the text to precisely produce all possible entity pairs in a sequential manner. Experiments on three benchmark datasets show that our proposed method successfully addresses the multiple relations and multiple entity pairs even with complex overlapping and significantly outperforms the state-of-the-art methods. All source code and documentations are available at https://github.com/chenjiayu1502/MrMep.

1 Introduction

Extracting relational facts from unstructured text is a significant step in building large-scale knowledge graphs. A relational fact is typically represented as a triplet $<h, r, t>$ where h represents a head entity, t represents a tail entity, and r is a relation that connects h to t.

A number of neural models for extracting relational facts have been developed in recent years, and the most successful models all have one thing in common: they extract both relation and its corresponding entity pairs in a manner of joint learning (Zheng et al., 2017; Zeng et al., 2018; Takanobu et al., 2019). However, jointly extracting relation and entity pairs is far from a trivial task due that there might exist more than one relation within the text, while each target relation might correspond to more than one entity pair. A more challenging aspect of this problem is that there might exist complex overlapping among different triplets. As exemplified in Figure 1, the entity "$Indonesia$" overlaps in all triplets within the text, while $<Indonesia, leaderName, Jusuf\ Kalla>$ and $<Indonesia, leaderName, Joko\ Widodo>$ have the same relation type "$leaderName$", $<Bakso, region, Indonesia>$ and $<Bakso, country, Indonesia>$ share the same head and tail entities.

A well-defined relational fact extraction task aims to detect all possible relation types in the text, and extract all candidate entity pairs for each target relation type, while taking into account the complicated overlapping among the triplets.

Zheng et al. (2017) propose a tagging mechanism (referred as Tagging thereafter) to transform the relational fact extraction into a sequence labeling task by injecting the information of relation type and entity position into tags. In this paradigm, Tagging model assigns a unique label to each word, which fails to extract triplets with overlapped entities or even overlapped entity pairs.

Zeng et al. (2018) model the triplet extraction using sequence-to-sequence learning with copy mechanism (referred as CopyR thereafter) that is often used in sentence generation (Gu et al., 2016; He et al., 2017). Takanobu et al. (2019) apply a hierarchical framework with reinforcement learning, which decomposes triplet extraction into a high-level task for relation detection and a low-level task for entity extraction (referred as HRL thereafter). Both CopyR and HRL determine a relation type each time and detect an entity pair for it, then

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 593–602
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

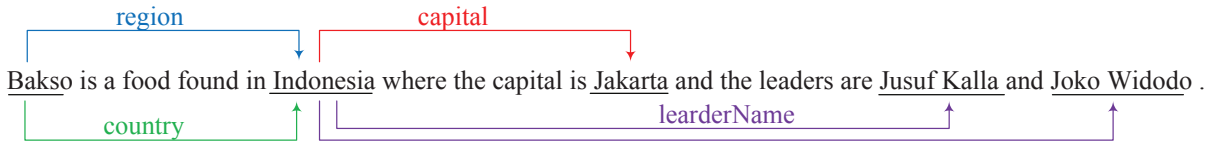


Figure 1: An example of multiple triplets. Entities are underlined, the relation types between entity pairs are connected by colored lines marked with relation types.

repeat the process to extract all triplets. However, to extract multiple entity pairs for a relation type, both CopyR and HRL have to repeatedly predict the relation type in multiple passes, which is computationally inefficient.

In this work, we propose, with simplicity and effectiveness in mind, a novel approach for jointly extracting Multiple Relations and Multiple Entity Pairs (MrMep). MrMep utilizes a triplet attention to exploit connections between relation and its corresponding entity pairs. It first predicts all possible relations, then for each target relation, it uses a variant of the pointer network (Vinyals et al., 2015) to generate boundaries (START/END positions) of all head and tail entities in a sequential manner, whereby the model generates all possible entity pairs as answers. In this way, for each candidate relation type, relation detection only needs to be performed one time, and then all possible entity pairs can be extracted for it, avoiding the repeated process of relation identification that is adopted in both CopyR and HRL. Moreover, we approach entity overlapping problems by a variant of the pointer network. It can sequentially generate entity boundaries in arbitrary position within the text. Therefore, it allows entities freely to participate in different triplets.

Our contributions can be summarized as follows:

- We propose MrMep, a novel neural method which firstly extracts all possible relations and then extracts all possible entity pairs for each target relation, while the two procedures are packed together into a joint model and are trained in a joint manner.
- MrMep uses a triplet attention to strengthen the connections among relation and entity pairs, and is computationally efficient for sophisticated overlapped triplets even with lightweight network architecture.
- Through extensive experiments on three benchmark datasets, we demonstrate MrMep's effectiveness over the most competitive state-of-the-art approach by 7.8%, 7.0% and 9.9% improvement respectively in F1 scores.

2 Related Work

Traditional pipelined approaches (Chan and Roth, 2011; Zelenko et al., 2003; Lin et al., 2016) separately design two subtasks of relation identification and entity pair extraction, ignoring the connection between them (Li and Ji, 2014) and suffering from error propagation.

To address the above problems, several joint models have been proposed. Early work (Li and Ji, 2014; Yu and Lam, 2010; Miwa and Sasaki, 2014) builds the connections between relation identification and entity recognition by designing multiple ingenious features, which needs complicated feature engineering. Recently, with the success of deep learning on many NLP tasks, several pieces of work (Miwa and Bansal, 2016; Gupta et al., 2016; Zhang et al., 2017) jointly model the interaction between the two subtasks based on neural networks, which they firstly apply RNN or CNN to encode the text, then treat entity recognition as a sequence labeling task and regard relation extraction as a multi-class classification problem. Zheng et al. (2017) formulate joint triplet extraction problem to the task of sequence labeling, by deliberately designing a tagging schema that injects information of relation type and entity position into tags. However, due to the limitation that a word can only be assigned a unique label in sequence labeling task and unique relation type can be assigned to each entity pair in multi-class classification problem, the above methods cannot fully address the triplet overlapping problem.

Most recent work aims at further exploring the overlapping triplet extraction. Zeng et al. (2018) propose CopyR, a joint model based on copy mechanism to convert the joint extraction task into a triplet generation task. For simplification, CopyR only copies the last word of the

entity, thereafter it cannot extract the whole entities consisting of multiple words. Takanobu et al. (2019) propose a hierarchical reinforcement learning framework that decomposes the joint extraction task into a high-level task for relation detection and a low-level task for entity extraction. However, both CopyR and HRL generate each relation type along with one entity pair at each time. In this paradigm, both of them have to perform the same relation detection multiple times to explore all possible entity pairs, which is computationally inefficient and principled not graceful enough.

We propose MrMep, a novel joint extraction framework that not only can jointly model triplet extraction task, but also efficiently extract various overlapped triplets. First, MrMep executes multiple relation classifiers by adopting a binary classifier for each relation type. Second, MrMep performs variable-length entity pair predictor. It utilizes a pointer network (Vinyals et al., 2015) alike way to generate the START/END positions for all the candidate head and tail entities in a sequential fashion, which is tracked by an LSTM decoder until it produces a word position beyond the text boundary. Inspired by machine reading and comprehending models (Wang and Jiang, 2016; Seo et al., 2017), we use the relation as a query, and conduct interaction between query and text via a triplet attention with relation peeking all possible entity pairs. We discuss two different principled ways to perform the above triplet attention, and demonstrate the versatility and effectiveness of our methods on both English and Chinese relational fact extraction benchmarks.

The main difference between our models and CopyR and HRL is that ours has the advantage of learning a strategic decoder using triplet attention that can model the interaction between relation and entity pairs. By letting the decoder only predict entity pairs, we relieve it from the burden of having to generate relation types repeatedly.

Das et al. (2019)'s work is similar in spirit to our work in that we both use a machine reading comprehension framework to extract relations and entities. But our approach differs from that of Das et al. (2019): their work aims at extracting newest state from procedural text which mainly describes the entity state at different steps, while our work is proposed for extracting multiple triplets from arbitrary unstructured text. Another similar approach is introduced by Roth et al. (2019), which performs relation argument extraction given a query entity and relation, essentially not for joint triplet extraction.

3 The Approach

3.1 Overview

For relational fact extraction task, the input is a text paragraph with n words $S = [w_1, ..., w_n]$. Let $\mathcal{R}$ be the set of predefined relation types. The task is to predict all possible triplets $<e_i, r_{ij}, e_j>$ maintained in S, where e_i, e_j are sequences of tokens denoting head entity and tail entity respectively, and $r_{ij} \in \mathcal{R}$ is the relation type that connects e_i to e_j.

Figure 2 shows an overview architecture of the proposed MrMep. It consists of three main parts: Encoder, Multiple Relation Classifiers and Variable-length Entity Pair Predictor. The encoder preprocesses the source text and extracts the sequence-level features using a Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), the multiple relation classifiers predict all possible relations maintained in S, and the variable-length entity pair predictor sequentially generates all possible entity pairs for each possible relation type.

3.2 Encoder

Given a text paragraph $S = [w_1, ..., w_n]$, we first embed it to obtain text embedding $E \in \mathbb{R}^{n \times d_e}$, where n is the length of words in text, d_e denotes the dimension of word embedding pretrained using Glove (Pennington et al., 2014). Then an LSTM is used to learn the token representation X_i for each word w_i:

$$X_i = LSTM_{encoder}(E_i, X_{i-1}), \quad (1)$$

where X_{i-1}, $X_i \in \mathbb{R}^d$ denotes word vectors for word w_{i-1} and w_i, yielding the text representation matrix $X = [X_1, ...X_n] \in \mathbb{R}^{n \times d}$.

The LSTM maps the variable length input sequences to a fixed-sized vector, and uses the last hidden state $X_n \in \mathbb{R}^d$ as the representation vector of the text.

3.3 Multiple Relation Classifiers

The relation classifier aims to identify relation types contained in the text. Since a text may contain multiple relations, inspired by the idea of multi-label classification, we design M CNN (Kim, 2014) based binary classifiers, respectively

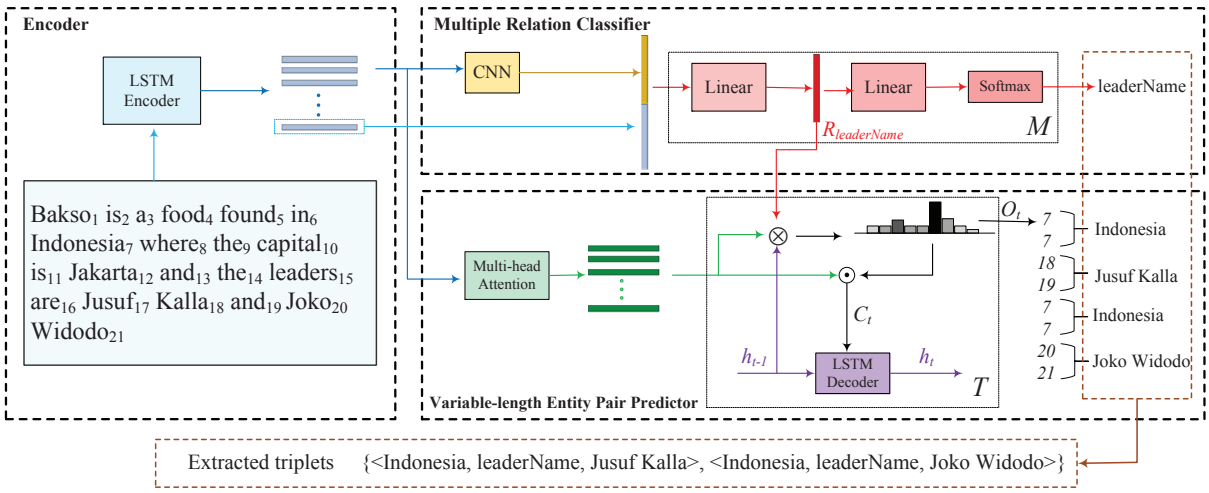


Figure 2: The model architecture of MrMep.

for M relation types, whose output is the probability distribution over whether the corresponding relation is a possible relation or not.

We adopt a convolutional neural network with the same structure as in Kim (2014). Given the text representation $X \in \mathbb{R}^{n \times d}$. A convolution operator and max-over-time pooling (Collobert et al., 2011) operator are applied:

$$C = Conv(X), \tag{2}$$

$$Q = relu(max(C)), \tag{3}$$

where $C \in \mathbb{R}^{m \times (n-l+1)}$ is the feature map output by the convolution operator, m is the number of different filters, n is the length of the text, l is the convolutional filter size. Eq. (3) applies an max-over-time pooling operator on feature map C first, and then a $relu$ activation is used to obtain the text embedding $Q \in \mathbb{R}^m$. When sliding over the text with a window size l, different filters offer a variety of l-gram compositions. Therefore, the text embedding Q is viewed as local feature vector of the text in our model.

In order to make better use of the features extracted by LSTM and CNN, a concatenation operator is used between the last word representation X_n in the encoder and text embedding Q from CNN to produce a fused vector $H \in \mathbb{R}^{m+d}$:

$$H = Concat(Q, X_n). \tag{4}$$

The binary classifier for j-th relation type is shown as follows (We omit the bias b for simplification):

$$R_j = HW_j^H, \tag{5}$$

$$Y^j = softmax(R_j W_j^R). \tag{6}$$

As in Eq. (5), a linear layer is applied to produce the hidden layer state $R_j \in \mathbb{R}^d$. $W_j^H \in \mathbb{R}^{(m+d) \times d}$ is a learnable weight matrix. As in Eq. (6), another linear layer with a $softmax$ activation function is used to predict the probability distribution of whether the text contains the j-th relation type or not, $Y^j \in \mathbb{R}^2$. $W_j^R \in \mathbb{R}^{d \times 2}$ is weight parameter. The hidden layer state R_j is viewed as the relation embedding for j-th relation type.

If the text contains j-th relation type, R_j will be fed into variable-length entity pair predictor to aid entity pair recognition.

3.4 Variable-length Entity Pair Predictor

Given a text, and a target relation type output by the relation classifier, the variable-length entity pair predictor aims to extract its all possible entity pairs in a sequence manner. Inspired by the pointer network (Vinyals et al., 2015), we determine an entity by identifying START and END position indexes of the words in the text. As shown in Figure 2, entity pairs are generated by a sequence of indexes. Each two indexes can identify an entity and each two entities form an entity pair in order. In this paradigm, our model can explore all the possible relations in one pass and predict all possible entity pairs for a given relation via a lightweight sequence decoder, unlike previous work which has to predict the target relation in a multi-pass way (Zeng et al., 2018; Takanobu et al., 2019).

Multi-head Attention. We apply a multi-head attention (Vaswani et al., 2017) on X to obtain a

new text representation P:

$$Q = XW_j^Q, K = XW_j^K, V = XW_j^V, \quad (7)$$

$$head_j = softmax(\frac{QK^T}{\sqrt{d_k}})V, \quad (8)$$

$$P = Concat(head_1, ...head_h)W^O, \quad (9)$$

where different linear layers are used in Eq. (7) to map X into different subspaces by learnable parameters W_j^Q, $W_j^K \in \mathbb{R}^{d \times d_k}$, and $W_j^V \in \mathbb{R}^{d \times d_v}$, yielding query $Q \in \mathbb{R}^{n \times d_k}$, keys $K \in \mathbb{R}^{n \times d_k}$ and values $V \in \mathbb{R}^{n \times d_v}$. By Scaled Dot-Product Attention (Vaswani et al., 2017), $head_j \in \mathbb{R}^{n \times d_v}$ is obtained as the representation of j-th head. A concatenation operator and a linear layer are used in Eq. (9) to produce text representation $P \in \mathbb{R}^{n \times d}$, where h is the number of heads and $W^O \in \mathbb{R}^{hd_v \times d}$ is learnable parameter.

Triplet Attention. At each position of the text, attention mechanism is used to obtain a weighted value that represents the matching degree between the token and the target relation type. Since its aim is to extract the candidate entity pair for the target relation, we call this attention the triplet attention. The triplet attention essentially aggregates the matching of the relation type to each token of the text and used the aggregated matching result to make the entity prediction.

Assume the j-th relation is the target relation identified by the relation classifier in Section 3.3. To get the final attention of j-th relation to i-th token, we study two different modes to implement the triplet attention: paralleled mode and layered mode.

Paralleled Mode. The so-called paralleled mode draws the connections among the target relation, the token, and previous hidden state in a synchronous way:

$$a_t^i = W^a tanh(W^r \circ R_j + W^d \circ d_{t-1} + W^p \circ P_i), \quad (10)$$

where $\circ$ is the element-wise multiplication operator. $R_j \in \mathbb{R}^d$ is j-th relation embedding, $P_i \in \mathbb{R}^d$ is i-th word of text representation P, $d_{t-1} \in \mathbb{R}^d$ is hidden state of LSTM decoder at time step $t-1$ which obtained by Eq. (13). W^r, W^d, W^p, and $W^a \in \mathbb{R}^d$ are learnable parameters. $a_t^i \in \mathbb{R}$ is attention weight of i-th word in the text.

The attention distribution on text $\alpha_t \in \mathbb{R}^n$ is computed as follows:

$$\alpha_t = softmax(a_t), \quad (11)$$

where $\alpha_t = [\alpha_t^1, ...\alpha_t^n]$, n denotes the text length. It is worthy to note that, α_t^i is used as the probability that position index i is selected as output at time step t. At time step t, the position index with the highest probability in α_t is greedily sampled as the output O_t.

Once the attention weights are computed, the context vector c_t is computed by:

$$c_t = \sum_{i=1}^{n} \alpha_t^i \cdot P_i. \quad (12)$$

Then c_t together with d_{t-1} are fed into LSTM decoder at time step t:

$$d_t = LSTM_{decoder}(c_t, d_{t-1}). \quad (13)$$

Therefore, the LSTM decoder is capable of tracking the state of the variable-length entity pair predictor.

Layered Mode. The layered mode first computes the connection between R_j and d_{t-1}:

$$\beta_t = tanh(W^{r'} \circ R_j + W^{d'} \circ d_{t-1}), \quad (14)$$

where W^β, $W^{p'}$, and $W^{a'} \in \mathbb{R}^d$ are learnable parameters.

Then vector β_t is used to calculate triplet attention:

$$a_i^t = W^{a'} tanh(W^\beta \circ \beta_t + W^{p'} \circ P_i), \quad (15)$$

where W^β, $W^{p'} \in \mathbb{R}^d$ are learnable parameters. The following procedure is conducted as the same denoted in Eq. (11)-(13).

3.5 Training

We adopt the cross-entropy loss function to define the loss of multiple relation classifiers and variable-length entity pair predictor respectively, which denoted as L_{rel} and L_{ent}.

To jointly train the model, the loss L is obtained by:

$$L = \lambda \cdot L_{rel} + (1 - \lambda) \cdot L_{ent}, \quad (16)$$

where $\lambda \in \mathbb{R}$ is a hyperparameter to balance the multiple relation classifiers and the variable-length entity pair predictor.

4 Experiments

4.1 Dataset

In order to test our proposed recipe for jointly extracting multiple relations and multiple entity

Dataset	NYT	WebNLG	SKE
Language	English	English	Chinese
Relation	24	246	50
Triplets	104,518	12,863	111,539
Token	90,760	5,051	170,206

Table 1: Statistics of three datasets in language, number of relation types, triplet number and token numbers.

pairs, we conducted experiments on two English benchmark datasets: New York Times (NYT) and WebNLG, and a Chinese public dataset Schema based Knowledge Extraction (SKE).

NYT is produced by distant supervision method (Riedel et al., 2010) and widely used in the triplet extraction (Zheng et al., 2017; Zeng et al., 2018). There are 24 relation types, 61,265 train texts and 5,000 test texts in NYT dataset. We randomly select 5,000 texts from train data as the development set.

WebNLG dataset (Gardent et al., 2017), is originally a dataset for natural language generation task and later used for triplet extraction (Zeng et al., 2018). There are 246 relation types, 5,321 train texts and 695 test texts. We randomly select 500 from train data as the development set.

Different from the two previous English datasets, SKE is a Chinese dataset for information extraction, which is released in the 2019 Language and Intelligence Challenge[1]. SKE contains 50 relation types and training texts exceed 200,000. We build our training set, development set, and test set by randomly selecting 50,000, 5,000 and 5,000 texts respectively.

Following the work of Zeng et al. (2018), we prepropose the three datasets as follows: (1) remove texts that contain no triplets at all; (2) filter out texts if there is an entity in the triplet that is not found in the text.

As shown in Table 1, SKE has a richest vocabulary (170,206 tokens), while WebNLG has a smallest vocabulary (5,051 tokens). In contrast to NYT and SKE which has a medium body of relations (24 and 50, respectively), WebNLG has a significantly big body of relations (246 relations).

4.2 Baseline and Evaluation Metrics

To evaluate our method, we compared against one baseline model and three state-of-the-art models.

[1]http://lic2019.ccf.org.cn/kg

Hyperparameter	value
dropout rate	0.5
learning rate	0.001
batch size	50
hidden size of LSTM	100
filter number of CNN	100
window size of CNN	3
head number	4
λ	0.3

Table 2: Hyperparameter setting.

Baseline: In our proposed baseline model, we design similar architecture with MrMep, in which the encoder and multiple relation classifiers are the same with MrMep, but we use Match-LSTM (Wang and Jiang, 2016) as an implementation of variable-length entity pair predictor. The three main layers of Match-LSTM in our baseline model are as follows: (1) LSTM Preprocessing Layer: we use output of the encoder as passage representation and relation embedding of multiple relation classifiers as the query representation; (2) Match-LSTM Layer: we make concatenation of query representation and each token embedding of passage representation to obtain query-aware token representation, and then fed it into a Bi-LSTM layer; (3) Answer Pointer Layer: this layer is same with Match-LSTM (Wang and Jiang, 2016) and we adopt the sequence model to produce entity pairs.

Tagging (Zheng et al., 2017): This model is a sequence labeling model which assigns to each token a unique tag denoting the information of relation, head or tail entity, even if that the token participates in two different triplets.

CopyR (Zeng et al., 2018): This model is a seq2seq model utilizing copy mechanism that generating a triplet by jointly copying a relation from relation set and copying an entity pair from the source texts in a sequential manner.

HRL (Takanobu et al., 2019): This model applies a hierarchical reinforcement learning framework that decomposes the task into a high-level task for relation detection and a low-level task for entity extraction, which is jointly optimized through a reinforcement learning paradigm.

Following (Zheng et al., 2017; Zeng et al., 2018; Takanobu et al., 2019), we use the standard micro Precision, Recall and F1 score to evaluate the results. Triplets are regarded as correct when the

Model	NYT			WebNLG			SKE		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Tagging	0.526	0.336	0.410	0.476	0.186	0.267	0.347	0.158	0.220
CopyR	0.602	0.556	0.578	0.381	0.369	0.375	0.402	0.362	0.380
HRL	0.741	0.651	0.693	**0.695**	0.629	0.660	0.582	0.422	0.489
Baseline	0.743	0.730	0.736	0.641	0.744	0.689	0.607	0.560	0.583
MrMep (para)	0.769	0.730	0.747	**0.695**	0.759	0.725	0.589	0.543	0.565
MrMep (layer)	**0.779**	**0.766**	**0.771**	0.694	**0.770**	**0.730**	**0.611**	**0.567**	**0.588**

Table 3: Results of different models in three datasets. MrMep (para) denotes that we adopt paralleled mode of triplet attention in variable-length entity pair predictor. MrMep (layer) denotes the layered mode.

Model	NYT10			NYT11			NYT10-sub			NYT11-plus		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
Tagging	0.593	0.381	0.464	0.469	0.489	0.479	0.256	0.237	0.246	0.292	0.220	0.250
CopyR	0.569	0.452	0.504	0.347	0.534	0.421	0.392	0.263	0.315	0.329	0.224	0.264
HRL	0.714	0.586	0.644	**0.538**	**0.538**	**0.538**	0.815	0.475	0.600	0.441	0.321	0.372
MrMep	**0.717**	**0.635**	**0.673**	0.434	0.522	0.474	**0.832**	**0.550**	**0.662**	**0.512**	**0.327**	**0.399**

Table 4: Detailed results of different models in datasets used in HRL (Takanobu et al., 2019). NYT10-sub is selected from NYT10 test set (original version), NYT11-plus is splitted from NYT-11 train set (manually created version). NYT10-sub and NYT11-plus contains a variety of overlapping triplets.

relation type and entity pair are both correct.

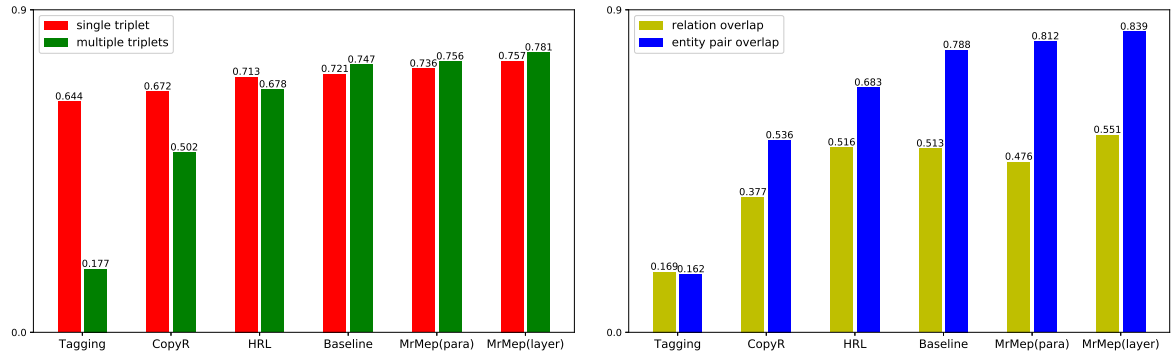
4.3 Implementation Details

We evaluate two variants of our approach: MrPep (para) using paralleled mode for triplet attention and MrMep (layer) using layered mode. For both variants, all hyperparameters are tuned on the same validation set. The hyperparameter setting is shown in Table 2. The word embedding are initialized using Glove (Pennington et al., 2014) and are updated during training, and the dimension is set to 100. The cell unit number of LSTM encoder and decoder is set to 100. The filter number used in CNN classifier is 100, filter window is 3, and the following dense layer has a hidden layer with 100 dimensions. The learning rate is set to 0.001. The tradeoff parameter λ in loss function is set to 0.3. The batch size is set to 50. The head number in muti-head attention is set to 4. For training, we use Adam (Kingma and Ba, 2015) to optimize parameters.

4.4 Main Results

Table 3 shows the Precision, Recall and F1 value of different models on the three datasets. When compared to Tagging and CopyR, the proposed model MrMep substantially improves the F1 scores in three datasets. Both the MrMep (para) and MrMep (layer) outperform the most competitive HRL model in three F1 scores. Moreover, compared with HRL, both the baseline model and our proposed models achieve a significant increase in Recall. Specifically, MrMep (layer) achieves 11.5% improvement in NYT, 14.1% improvement in WebNLG, and 14.5% improvement in SKE in Recall value. Our hypothesis is that the state-of-the-art models such as CopyR and HRL are seeking connection between relation and entity pairs in a sequential fashion, whereas we seek this connection in an interactive manner with the relation paying attention to the tokens one by one interactively.

To draw a paralleled comparison with HRL in more details, we compare the results on the same data partition used in HRL: NYT10, NYT11, NYT10-sub and NYT11-plus. The results of different approaches are summarized in Table 4. On NYT10, NYT10-sub and NYT11-plus test set, MrMep (layer) achieves the highest F1 values when compared with three state-of-the-art models. In NYT11 test set, our model gets poorer performance than HRL. But it is worthy to note that NYT11 dataset averagely contains a triplet per text (totally 369 texts with 370 triplets), while NYT10-sub and NYT11-plus test set contain more overlapping triplets. These observations suggest that our model is essentially feasible to extract more complicated triplets.



(a) (b)

Figure 3: Detailed results on different test subsets. The ordinate of the coordinate axis represents the F1 value.

Model	NYT	WebNLG	SKE
MrMep	**0.771**	**0.730**	**0.588**
w/o CNN	0.745	0.724	0.530
w/o Multi-head	0.723	0.616	0.512

Table 5: An ablation study for MrMep (layer) model. All values are F1 scores.

4.5 Detailed Results

To further verify the ability of MrMep in handling the various complex triplets, we conduct a series of qualitative analysis from two distinct views:

Single triplet vs Multiple triplets. According to the number of triplets contained in the text, we divide the NYT test set into two sub-datasets: (1) single triplet test set: each text contains only one triplet, and 3,240 texts are in the set; (2) multi triplet test set: each text contains two or more triplets, and 1,760 texts are in the set.

Relation overlap vs entity pair overlap. According to whether an overlap is relation overlap or entity pair overlap, we obtain two subsets from the test set: (1) relation overlap: one relation connects with two or more different entity pairs (462 texts in NYT test set); (2) entity pair overlap: one entity pair connects with two or more relations (969 texts in NYT test set).

Figure 3 (a) shows the F1 values of different approaches on NYT test data with single triplet and multiple triplets. It can be seen that both MrMep (para) and MrMep (layer) outperform the baseline and the three comparative models. Noticeably, the improvements on multiple triplets are more remarkable.

From Figure 3 (b) we can observe that MrMep (layer) outperforms MrMep (para) and both of them outperform the reference models. It is also interesting to notice that predicting relation overlaps is more challenging than predicting entity pair overlaps.

The baseline model adopts Match-LSTM (Wang and Jiang, 2016) as an implement of entity pair predictor, which sequentially aggregates the matching of the attention-weighted query to each token of the text. Although we could treat the relation as a query, the relation is essentially a tag with much simpler semantics than that of actual queries. Therefore, we design a lightweight MrMep model which is proven to be more elegant and powerful for entity pair extraction.

4.6 Ablation Study

We examine the contributions of two main components, namely, convolutional neural network (CNN) in multiple relation classifiers and multi-head attention in entity pair predictor, using the best-performing MrMep model with layered mode on three dataset. First, instead of using representations learnt by CNN, we use the last hidden state of the LSTM encoder as text presentation, then feed it directly to the multiple binary classifier (MrMep w/o CNN). Second, instead of applying multi-head attention, we use the hidden state at each time step of the LSTM encoder to represent each token, which is directly fed into the following triplet attention to extract entity pairs (MrMep w/o Multi-head). Table 5 shows the results. We can observe that adding either CNN or multi-head attention improves the performance of the model. This suggests that both parts can assist MrMep to jointly extract entities and relations, where the CNN layer seems to be playing a more significant role. One possible reason is that CNN is used for

MrMep (layer)	(NYT): "It 's pretty neat", said Ward, who grew up in the [Edmonton]$_{E_t-contains}$ suburb of Sherwood Park, [Alberta]$_{E_h-contains}$.
MrMep (para)	(NYT): "It 's pretty neat", said Ward, who grew up in the [Edmonton suburb of Sherwood Park]$_{E_t-contains}$, [Alberta]$_{E_h-contains}$.
MrMep (layer)	(WebNLG): [Bakso]$_{E_h-country}$ comes from [Indonesia]$_{E_t-country}$, and celery is a main ingredient. [Jasuf Kalla]$_{E_t-leaderName}$ is a leader in [Indonesia]$_{E_h-leaderName}$.
MrMep (para)	(WebNLG): [Bakso]$_{E_h-country}$ comes from [Indoneisa]$_{E_t-country}$, and [celery is a main ingredient. Jasuf Kalla]$_{E_t-leaderName}$ is a leader in [Indonesia]$_{E_h-leaderName}$.

Table 6: Extracted results from MrMep (para) and MrMep (layer). The words in a bracket represents an entity, E_h stands for the head entity and E_t stands for the tail entity. Red is marked as identifying the wrong.

extracting local feature of input text, which is not only beneficial for relation extraction but also assist producing better relation embedding to aid entity pair predition.

4.7 Case Study

Table 6 shows two examples of extracted results from MrMep (para) and MrMep (layer). In the first example, MrMep (layer) precisely extracts the correct triplet <Alberta, contains, Edmonton>, while MrMep (para) recognizes incorrect tail entity " Edmonton suburb of Sherwood Park". In the second example, MrMep (layer) precisely extracts correct triplet <Indoneisa, leaderName, Jasuf Kalla> while MrMep (para) wrongly recognizes the tail entity " celery is a main ingredient. Jasuf Kalla". Taking NYT test set as example, among 1250 differences between the output of MrMep (para) and MrMep (layer), there are 885 differences resulting from the wrong entity boundaries identified by MrMep (para). The most likely reason might be that the hidden state of the decoder plays as crucial role as the relation in sequentially producing different entity pairs.

5 Conclusion

We propose a joint multiple relation and multiple entity pair extraction model. The model uses a triplet attention to model the connection of relation and entities in a novel and lightweight framework. It gives state-of-the-art performance on three benchmark datasets.

References

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 551–560. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2019. Building dynamic knowledge graphs from text using machine reading comprehension. *In Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 199–208.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *Proceedings of ICLR*, pages 1–15.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 402–412.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2124–2133.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1116, Berlin, Germany. Association for Computational Linguistics.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Benjamin Roth, Costanza Conforti, Nina Poerner, Sanjeev Kumar Karn, and Hinrich Schütze. 2019. Neural architectures for open-type relation argument extraction. *Natural Language Engineering*, 25(2):219–238.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ICLR 2017*.

Ryuichi Takanobu, Tianyang Zhang, Jiexi Liu, and Minlie Huang. 2019. A hierarchical framework for relation extraction with reinforcement learning. *AAAI 2019*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1399–1407. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of machine learning research*, 3(Feb):1083–1106.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2017. End-to-end neural relation extraction with global optimization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1730–1740.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, Vancouver, Canada. Association for Computational Linguistics.

Effective Attention Modeling for Neural Relation Extraction

Tapas Nayak and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
nayakt@u.nus.edu, nght@comp.nus.edu.sg

Abstract

Relation extraction is the task of determining the relation between two entities in a sentence. Distantly-supervised models are popular for this task. However, sentences can be long and two entities can be located far from each other in a sentence. The pieces of evidence supporting the presence of a relation between two entities may not be very direct, since the entities may be connected via some indirect links such as a third entity or via co-reference. Relation extraction in such scenarios becomes more challenging as we need to capture the long-distance interactions among the entities and other words in the sentence. Also, the words in a sentence do not contribute equally in identifying the relation between the two entities. To address this issue, we propose a novel and effective attention model which incorporates syntactic information of the sentence and a multi-factor attention mechanism. Experiments on the New York Times corpus show that our proposed model outperforms prior state-of-the-art models.

1 Introduction

Relation extraction from unstructured text is an important task to build knowledge bases (KB) automatically. Banko et al. (2007) used open information extraction (Open IE) to extract relation triples from sentences where verbs were considered as the relation, whereas supervised information extraction systems extract a set of pre-defined relations from text. Mintz et al. (2009), Riedel et al. (2010), and Hoffmann et al. (2011) proposed distant supervision to generate the training data for sentence-level relation extraction, where relation tuples (two entities and the relation between them) from a knowledge base such as Freebase (Bollacker et al., 2008) were mapped to free text (Wikipedia articles or New York Times articles). The idea is that if a sentence contains both entities of a tuple, it is chosen as a training sentence of that tuple. Although this process can generate some noisy training instances, it can give a significant amount of training data which can be used to build supervised models for this task.

Mintz et al. (2009), Riedel et al. (2010), and Hoffmann et al. (2011) proposed feature-based learning models and used entity tokens and their nearby tokens, their part-of-speech tags, and other linguistic features to train their models. Recently, many neural network-based models have been proposed to avoid feature engineering. Zeng et al. (2014, 2015) used convolutional neural networks (CNN) with max-pooling to find the relation between two given entities. Though these models have been shown to perform reasonably well on distantly supervised data, they sometimes fail to find the relation when sentences are long and entities are located far from each other. CNN models with max-pooling have limitations in understanding the semantic similarity of words with the given entities and they also fail to capture the long-distance dependencies among the words and entities such as co-reference. In addition, all the words in a sentence may not be equally important in finding the relation and this issue is more prominent in long sentences. Prior CNN-based models have limitations in identifying the multiple important factors to focus on in sentence-level relation extraction.

To address this issue, we propose a *novel* multi-factor attention model[1] focusing on the syntactic structure of a sentence for relation extraction. We use a dependency parser to obtain the syntactic structure of a sentence. We use a linear form of attention to measure the semantic similarity of words with the given entities and combine

[1]The code and data of this paper can be found at https://github.com/nusnlp/MFA4RE

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 603–612
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

it with the dependency distance of words from the given entities to measure their influence in identifying the relation. Also, single attention may not be able to capture all pieces of evidence for identifying the relation due to normalization of attention scores. Thus we use multi-factor attention in the proposed model. Experiments on the New York Times (NYT) corpus show that the proposed model outperforms prior work in terms of F1 scores on sentence-level relation extraction.

2 Task Description

Sentence-level relation extraction is defined as follows: Given a sentence S and two entities $\{E_1, E_2\}$ marked in the sentence, find the relation $r(E_1, E_2)$ between these two entities in S from a pre-defined set of relations $R \cup \{None\}$. *None* indicates that none of the relations in R holds between the two marked entities in the sentence. The relation between the entities is argument order-specific, i.e., $r(E_1, E_2)$ and $r(E_2, E_1)$ are not the same. Input to the system is a sentence S and two entities E_1 and E_2, and output is the relation $r(E_1, E_2) \in R \cup \{None\}$.

3 Model Description

We use four types of embedding vectors in our model: (1) word embedding vector $\mathbf{w} \in \mathbb{R}^{d_w}$ (2) entity token indicator embedding vector $\mathbf{z} \in \mathbb{R}^{d_z}$, which indicates if a word belongs to entity 1, entity 2, or does not belong to any entity (3) a positional embedding vector $\mathbf{u}^1 \in \mathbb{R}^{d_u}$ which represents the linear distance of a word from the start token of entity 1 (4) another positional embedding vector $\mathbf{u}^2 \in \mathbb{R}^{d_u}$ which represents the linear distance of a word from the start token of entity 2.

We use a bi-directional long short-term memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) layer to capture the interaction among words in a sentence $S = \{w_1, w_2, ..., w_n\}$, where n is the sentence length. The input to this layer is the concatenated vector $\mathbf{x} \in \mathbb{R}^{d_w+d_z}$ of word embedding vector $\mathbf{w}$ and entity token indicator embedding vector $\mathbf{z}$.

$$\mathbf{x}_t = \mathbf{w}_t \mid\mid \mathbf{z}_t$$
$$\overrightarrow{\mathbf{h}_t} = \overrightarrow{\text{LSTM}}(\mathbf{x}_t, \mathbf{h}_{t-1})$$
$$\overleftarrow{\mathbf{h}_t} = \overleftarrow{\text{LSTM}}(\mathbf{x}_t, \mathbf{h}_{t+1})$$
$$\mathbf{h}_t = \overrightarrow{\mathbf{h}_t} || \overleftarrow{\mathbf{h}_t}$$

$\overrightarrow{\mathbf{h}_t} \in \mathbb{R}^{d_w+d_z}$ and $\overleftarrow{\mathbf{h}_t} \in \mathbb{R}^{d_w+d_z}$ are the output at the tth step of the forward LSTM and backward LSTM respectively. We concatenate them to obtain the tth Bi-LSTM output $\mathbf{h}_t \in \mathbb{R}^{2(d_w+d_z)}$.

3.1 Global Feature Extraction

We use a convolutional neural network (CNN) to extract the sentence-level global features for relation extraction. We concatenate the positional embeddings $\mathbf{u}^1$ and $\mathbf{u}^2$ of words with the hidden representation of the Bi-LSTM layer and use the convolution operation with max-pooling on concatenated vectors to extract the global feature vector.

$$\mathbf{q}_t = \mathbf{h}_t \| \mathbf{u}_t^1 \| \mathbf{u}_t^2$$
$$c_i = \mathbf{f}^T(\mathbf{q}_i \| \mathbf{q}_{i+1} \| .... \| \mathbf{q}_{i+k-1})$$
$$c_{max} = \max(c_1, c_2, ..., c_n)$$
$$\mathbf{v}_g = [c_{max}^1, c_{max}^2, ..., c_{max}^{f_g}]$$

$\mathbf{q}_t \in \mathbb{R}^{2(d_w+d_z+d_u)}$ is the concatenated vector for the tth word and $\mathbf{f}$ is a convolutional filter vector of dimension $2k(d_w + d_z + d_u)$ where k is the filter width. The index i moves from 1 to n and produces a set of scalar values $\{c_1, c_2, ..., c_n\}$. The max-pooling operation chooses the maximum c_{max} from these values as a feature. With f_g number of filters, we get a global feature vector $\mathbf{v}_g \in \mathbb{R}^{f_g}$.

3.2 Attention Modeling

Figure 1 shows the architecture of our attention model. We use a linear form of attention to find the semantically meaningful words in a sentence with respect to the entities which provide the pieces of evidence for the relation between them. Our attention mechanism uses the entities as attention queries and their vector representation is very important for our model. Named entities mostly consist of multiple tokens and many of them may not be present in the training data or their frequency may be low. The nearby words of an entity can give significant information about the entity. Thus we use the tokens of an entity and its nearby tokens to obtain its vector representation. We use the convolution operation with max-pooling in the context of an entity to get its vector representation.

$$c_i = \mathbf{f}^T(\mathbf{x}_i \| \mathbf{x}_{i+1} \| .... \| \mathbf{x}_{i+k-1})$$
$$c_{max} = \max(c_b, c_{b+1}, ..., c_e)$$
$$\mathbf{v}_e = [c_{max}^1, c_{max}^2, ..., c_{max}^{f_e}]$$

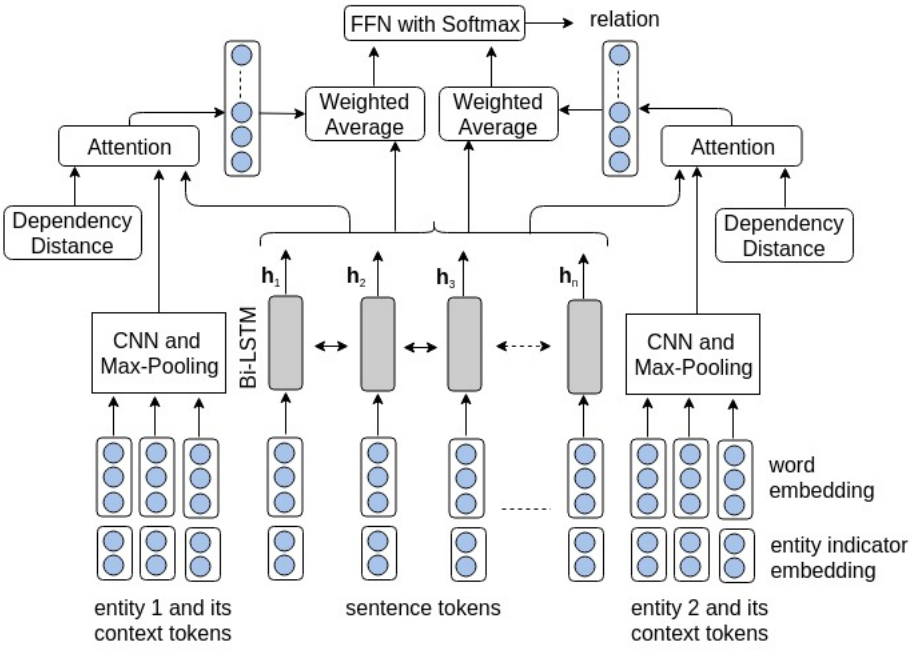


Figure 1: Architecture of our attention model with $m = 1$. We have not shown the CNN-based global feature extraction here. FFN=feed-forward network.

$\mathbf{f}$ is a convolutional filter vector of size $k(d_w + d_z)$ where k is the filter width and $\mathbf{x}$ is the concatenated vector of word embedding vector ($\mathbf{w}$) and entity token indicator embedding vector ($\mathbf{z}$). b and e are the start and end index of the sequence of words comprising an entity and its neighboring context in the sentence, where $1 \leq b \leq e \leq n$. The index i moves from b to e and produces a set of scalar values $\{c_b, c_{b+1}, ..., c_e\}$. The max-pooling operation chooses the maximum c_{max} from these values as a feature. With f_e number of filters, we get the entity vector $\mathbf{v}_e \in \mathbb{R}^{f_e}$. We do this for both entities and get $\mathbf{v}_e^1 \in \mathbb{R}^{f_e}$ and $\mathbf{v}_e^2 \in \mathbb{R}^{f_e}$ as their vector representation. We adopt a simple linear function as follows to measure the semantic similarity of words with the given entities:

$$\mathrm{f}^1_{\mathrm{score}}(\mathbf{h}_i, \mathbf{v}_e^1) = \mathbf{h}_i^T \mathbf{W}_a^1 \mathbf{v}_e^1$$
$$\mathrm{f}^2_{\mathrm{score}}(\mathbf{h}_i, \mathbf{v}_e^2) = \mathbf{h}_i^T \mathbf{W}_a^2 \mathbf{v}_e^2$$

$\mathbf{h}_i$ is the Bi-LSTM hidden representation of the ith word. $\mathbf{W}_a^1, \mathbf{W}_a^2 \in \mathbb{R}^{2(d_w+d_z)\times f_e}$ are trainable weight matrices. $\mathrm{f}^1_{\mathrm{score}}(\mathbf{h}_i, \mathbf{v}_e^1)$ and $\mathrm{f}^2_{\mathrm{score}}(\mathbf{h}_i, \mathbf{v}_e^2)$ represent the semantic similarity score of the ith word and the two given entities.

Not all words in a sentence are equally important in finding the relation between the two entities. The words which are closer to the entities are generally more important. To address this issue, we propose to incorporate the syntactic structure of a sentence in our attention mechanism. The syntactic structure is obtained from the dependency parse tree of the sentence. Words which are closer to the entities in the dependency parse tree are more relevant to finding the relation. In our model, we define the dependency distance to every word from the head token (last token) of an entity as the number of edges along the dependency path (See Figure 2 for an example). We use a distance window size ws and words whose dependency distance is within this window receive attention and the other words are ignored. The details of our attention mechanism follow.

$$d_i^1 = \begin{cases} \frac{1}{2^{l_i^1 - 1}} \exp(\mathrm{f}^1_{\mathrm{score}}(\mathbf{h}_i, \mathbf{v}_e^1)) & \text{if } l_i^1 \in [1, ws] \\ \frac{1}{2^{ws}} \exp(\mathrm{f}^1_{\mathrm{score}}(\mathbf{h}_i, \mathbf{v}_e^1)) & \text{otherwise} \end{cases}$$

$$d_i^2 = \begin{cases} \frac{1}{2^{l_i^2 - 1}} \exp(\mathrm{f}^2_{\mathrm{score}}(\mathbf{h}_i, \mathbf{v}_e^2)) & \text{if } l_i^2 \in [1, ws] \\ \frac{1}{2^{ws}} \exp(\mathrm{f}^2_{\mathrm{score}}(\mathbf{h}_i, \mathbf{v}_e^2)) & \text{otherwise} \end{cases}$$

$$p_i^1 = \frac{d_i^1}{\sum_j d_j^1}, \qquad p_i^2 = \frac{d_i^2}{\sum_j d_j^2}$$

d_i^1 and d_i^2 are un-normalized attention scores and p_i^1 and p_i^2 are the normalized attention scores for the ith word with respect to entity 1 and entity 2 respectively. l_i^1 and l_i^2 are the dependency distances

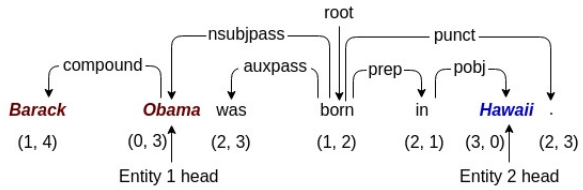


Figure 2: An example dependency tree. The two numbers indicate the distance of the word from the head token of the two entities respectively along the dependency tree path.

of the ith word from the two entities. We mask those words whose average dependency distance from the two entities is larger than ws. We use the semantic meaning of the words and their dependency distance from the two entities together in our attention mechanism. The attention feature vectors $\mathbf{v}_a^1$ and $\mathbf{v}_a^1$ with respect to the two entities are determined as follows:

$$\mathbf{v}_a^1 = \sum_{i=1}^{n} p_i^1 \mathbf{h}_i, \qquad \mathbf{v}_a^2 = \sum_{i=1}^{n} p_i^2 \mathbf{h}_i$$

3.3 Multi-Factor Attention

Two entities in a sentence, when located far from each other, can be linked via more than one coreference chain or more than one important word. Due to the normalization of the attention scores as described above, single attention cannot capture all relevant information needed to find the relation between two entities. Thus we use a multi-factor attention mechanism, where the number of factors is a hyper-parameter, to gather all relevant information for identifying the relation. We replace the attention matrix $\mathbf{W}_a$ with an attention tensor $\mathbf{W}_a^{1:m} \in \mathbb{R}^{m \times 2(d_w+d_z) \times 2f_e}$ where m is the factor count. This gives us m attention vectors with respect to each entity. We concatenate all the feature vectors obtained using these attention vectors to get the multi-attentive feature vector $\mathbf{v}_{ma} \in \mathbb{R}^{4m(d_w+d_z)}$.

3.4 Relation Extraction

We concatenate $\mathbf{v}_g$, $\mathbf{v}_{ma}$, $\mathbf{v}_e^1$, and $\mathbf{v}_e^2$, and this concatenated feature vector is given to a feed-forward layer with softmax activation to predict the normalized probabilities for the relation labels.

$$\mathbf{r} = \text{softmax}(\mathbf{W}_r(\mathbf{v}_g \mid\mid \mathbf{v}_{ma} \mid\mid \mathbf{v}_e^1 \mid\mid \mathbf{v}_e^2) + \mathbf{b}_r)$$

$\mathbf{W}_r \in \mathbb{R}^{(f_g+2f_e+4m(d_w+d_z)) \times (|R|+1)}$ is the weight matrix, $\mathbf{b}_r \in \mathbb{R}^{|R|+1}$ is the bias vector of the feed-forward layer for relation extraction, and $\mathbf{r}$ is the vector of normalized probabilities of relation labels.

3.5 Loss Function

We calculate the loss over each mini-batch of size B. We use the following negative log-likelihood as our objective function for relation extraction:

$$\mathcal{L} = -\frac{1}{B} \sum_{i=1}^{B} \log(p(r_i|s_i, e_i^1, e_i^2, \theta))$$

where $p(r_i|s_i, e_i^1, e_i^2, \theta)$ is the conditional probability of the true relation r_i when the sentence s_i, two entities e_i^1 and e_i^2, and the model parameters θ are given.

4 Experiments

4.1 Datasets

We use the New York Times (NYT) corpus (Riedel et al., 2010) in our experiments. There are two versions of this corpus: (1) The original NYT corpus created by Riedel et al. (2010) which has 52 valid relations and a *None* relation. We name this dataset NYT10. The training dataset has $455,412$ instances and $330,776$ of the instances belong to the *None* relation and the remaining $124,636$ instances have valid relations. The test dataset has $172,415$ instances and $165,974$ of the instances belong to the *None* relation and the remaining $6,441$ instances have valid relations. Both the training and test datasets have been created by aligning Freebase (Bollacker et al., 2008) tuples to New York Times articles. (2) Another version created by Hoffmann et al. (2011) which has 24 valid relations and a *None* relation. We name this dataset NYT11. The corresponding statistics for NYT11 are given in Table 1. The training dataset is created by aligning Freebase tuples to NYT articles, but the test dataset is manually annotated.

4.2 Evaluation Metrics

We use precision, recall, and F1 scores to evaluate the performance of models on relation extraction after removing the *None* labels. We use a confidence threshold to decide if the relation of a test instance belongs to the set of relations R or *None*. If the network predicts *None* for a test instance, then it is considered as *None* only. But if the network predicts a relation from the set R and the corresponding softmax score is below the confidence threshold, then the final class is changed to *None*.

		NYT10	NYT11
	# relations	53	25
Train	# instances	455,412	335,843
	# valid relation tuples	124,636	100,671
	# None relation tuples	330,776	235,172
	avg. sentence length	41.1	37.2
	avg. distance between entity pairs	12.8	12.2
Test	# instances	172,415	1,450
	# valid relation tuples	6,441	520
	# None relation tuples	165,974	930
	avg. sentence length	41.7	39.7
	avg. distance between entity pairs	13.1	11.0

Table 1: Statistics of the NYT10 and NYT11 dataset.

This confidence threshold is the one that achieves the highest F1 score on the validation dataset. We also include the precision-recall curves for all the models.

4.3 Parameter Settings

We run word2vec (Mikolov et al., 2013) on the NYT corpus to obtain the initial word embeddings with dimension $d_w = 50$ and update the embeddings during training. We set the dimension of entity token indicator embedding vector $d_z = 10$ and positional embedding vector $d_u = 5$. The hidden layer dimension of the forward and backward LSTM is 60, which is the same as the dimension of input word representation vector $\mathbf{x}$. The dimension of Bi-LSTM output is 120. We use $f_g = f_e = 230$ filters of width $k = 3$ for feature extraction whenever we apply the convolution operation. We use dropout in our network with a dropout rate of 0.5, and in convolutional layers, we use the tanh activation function. We use the sequence of tokens starting from 5 words before the entity to 5 words after the entity as its context. We train our models with mini-batch size of 50 and optimize the network parameters using the Adagrad optimizer (Duchi et al., 2011). We use the dependency parser from spaCy[2] to obtain the dependency distance of the words from the entities and use $ws = 5$ as the window size for dependency distance-based attention.

4.4 Comparison to Prior Work

We compare our proposed model with the following state-of-the-art models.

(1) CNN (Zeng et al., 2014): Words are represented using word embeddings and two positional embeddings. A convolutional neural network (CNN) with max-pooling is applied to extract the sentence-level feature vector. This feature vector is passed to a feed-forward layer with softmax to classify the relation.

(2) PCNN (Zeng et al., 2015): Words are represented using word embeddings and two positional embeddings. A convolutional neural network (CNN) is applied to the word representations. Rather than applying a global max-pooling operation on the entire sentence, three max-pooling operations are applied on three segments of the sentence based on the location of the two entities (hence the name Piecewise Convolutional Neural Network (PCNN)). The first max-pooling operation is applied from the beginning of the sentence to the end of the entity appearing first in the sentence. The second max-pooling operation is applied from the beginning of the entity appearing first in the sentence to the end of the entity appearing second in the sentence. The third max-pooling operation is applied from the beginning of the entity appearing second in the sentence to the end of the sentence. Max-pooled features are concatenated and passed to a feed-forward layer with softmax to determine the relation.

(3) Entity Attention (EA) (Shen and Huang, 2016): This is the combination of a CNN model and an attention model. Words are represented using word embeddings and two positional embeddings. A CNN with max-pooling is used to extract global features. Attention is applied with respect to the two entities separately. The vector representation of every word is concatenated with the word embedding of the last token of the entity. This concatenated representation is passed to a feed-forward layer with tanh activation and then another feed-forward layer to get a scalar attention score for every word. The original word representations are averaged based on the attention scores to get the attentive feature vectors. A CNN-extracted feature vector and two attentive feature vectors with respect to the two entities are concatenated and passed to a feed-forward layer with softmax to determine the relation.

(4) BiGRU Word Attention (BGWA) (Jat et al., 2017): Words are represented using word embeddings and two positional embeddings. They are passed to a bidirectional gated recurrent unit (BiGRU) (Cho et al., 2014) layer. Hidden vectors of the BiGRU layer are passed to a bilinear operator (a combination of two feed-forward layers)

[2]https://spacy.io/

Model	NYT10 Prec.	NYT10 Rec.	NYT10 F1	NYT11 Prec.	NYT11 Rec.	NYT11 F1
CNN (Zeng et al., 2014)	0.413	0.591	0.486	0.444	0.625	0.519
PCNN (Zeng et al., 2015)	0.380	**0.642**	0.477	0.446	0.679	0.538†
EA (Shen and Huang, 2016)	0.443	0.638	0.523†	0.419	0.677	0.517
BGWA (Jat et al., 2017)	0.364	0.632	0.462	0.417	**0.692**	0.521
BiLSTM-CNN	0.490	0.507	0.498	0.473	0.606	0.531
Our model	**0.541**	0.595	**0.566***	**0.507**	0.652	**0.571***

Table 2: Performance comparison of different models on the two datasets. * denotes a statistically significant improvement over the previous best state-of-the-art model with $p < 0.01$ under the bootstrap paired t-test. † denotes the previous best state-of-the-art model.

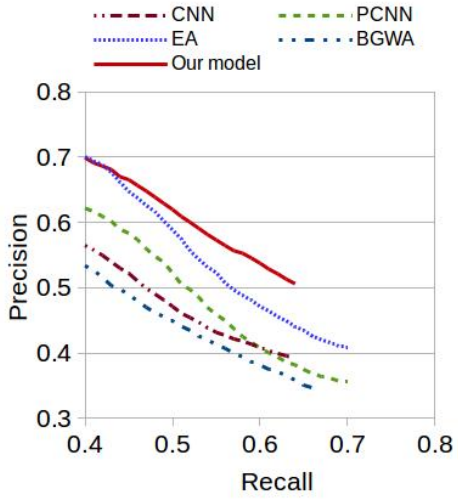


Figure 3: Precision-Recall curve for the NYT10 dataset.

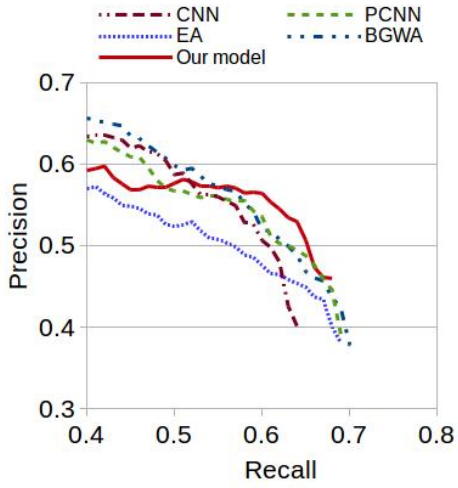


Figure 4: Precision-Recall curve for the NYT11 dataset.

to compute a scalar attention score for each word. Hidden vectors of the BiGRU layer are multiplied by their corresponding attention scores. A piecewise CNN is applied on the weighted hidden vectors to obtain the feature vector. This feature vector is passed to a feed-forward layer with softmax to determine the relation.

(5) BiLSTM-CNN: This is our own baseline. Words are represented using word embeddings and entity indicator embeddings. They are passed to a bidirectional LSTM. Hidden representations of the LSTMs are concatenated with two positional embeddings. We use CNN and max-pooling on the concatenated representations to extract the feature vector. Also, we use CNN and max-pooling on the word embeddings and entity indicator embeddings of the context words of entities to obtain entity-specific features. These features are concatenated and passed to a feed-forward layer to determine the relation.

4.5 Experimental Results

We present the results of our final model on the relation extraction task on the two datasets in Table 2. Our model outperforms the previous state-of-the-art models on both datasets in terms of F1 score. On the NYT10 dataset, it achieves 4.3% higher F1 score compared to the previous best state-of-the-art model EA. Similarly, it achieves 3.3% higher F1 score compared to the previous best state-of-the-model PCNN on the NYT11 dataset. Our model improves the precision scores on both datasets with good recall scores. This will help to build a cleaner knowledge base with fewer false positives. We also show the precision-recall curves for the NYT10 and NYT11 datasets in Figures 3 and 4 respectively. The goal of any relation extraction system is to extract as many relations as possible with minimal false positives. If the recall score becomes very low, the coverage of

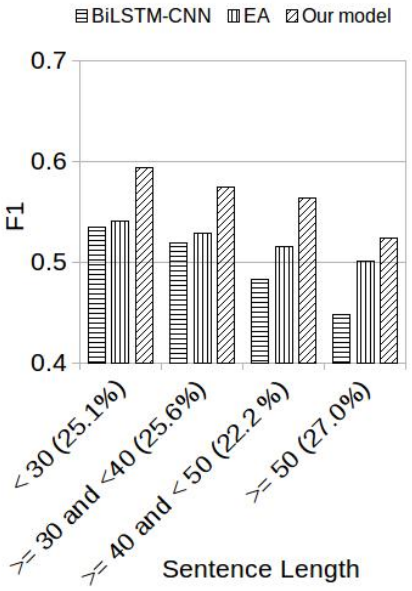


Figure 5: Performance comparison across different sentence lengths on the NYT10 dataset.

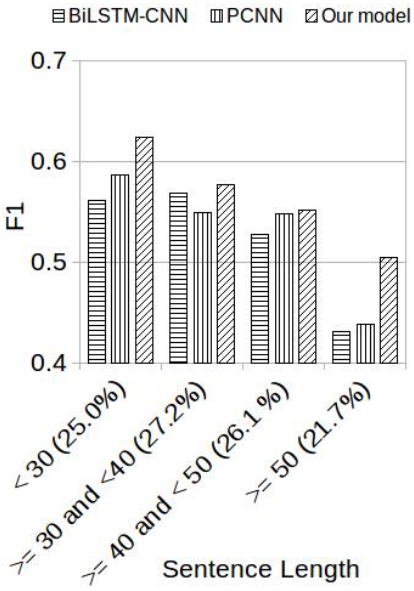


Figure 6: Performance comparison across different sentence lengths on the NYT11 dataset.

the KB will be poor. From Figure 3, we observe that when the recall score is above 0.4, our model achieves higher precision than all the competing models on the NYT10 dataset. On the NYT11 dataset (Figure 4), when recall score is above 0.6, our model achieves higher precision than the competing models. Achieving higher precision with high recall score helps to build a cleaner KB with good coverage.

5 Analysis and Discussion

5.1 Varying the number of factors (m)

We investigate the effects of the multi-factor count (m) in our final model on the test datasets in Table 3. We observe that for the NYT10 dataset, $m = \{1, 2, 3\}$ gives good performance with $m = 1$ achieving the highest F1 score. On the NYT11 dataset, $m = 4$ gives the best performance. These experiments show that the number of factors giving the best performance may vary depending on the underlying data distribution.

5.2 Effectiveness of Model Components

We include the ablation results on the NYT11 dataset in Table 4. When we add multi-factor attention to the baseline BiLSTM-CNN model without the dependency distance-based weight factor in the attention mechanism, we get 0.8% F1 score improvement (A2$-$A1). Adding the dependency weight factor with a window size of 5 improves

	NYT10			NYT11		
m	Prec.	Rec.	F1	Prec.	Rec.	F1
1	0.541	0.595	**0.566**	0.495	0.621	0.551
2	0.521	0.597	0.556	0.482	0.656	0.555
3	0.490	0.617	0.547	0.509	0.633	0.564
4	0.449	0.623	0.522	0.507	0.652	**0.571**
5	0.467	0.609	0.529	0.488	0.677	0.567

Table 3: Performance comparison of our model with different values of m on the two datasets.

the F1 score by 3.2% (A3$-$A2). Increasing the window size to 10 reduces the F1 score marginally (A3$-$A4). Replacing the attention normalizing function with softmax operation also reduces the F1 score marginally (A3$-$A5). In our model, we concatenate the features extracted by each attention layer. Rather than concatenating them, we can apply max-pooling operation across the multiple attention scores to compute the final attention scores. These max-pooled attention scores are used to obtain the weighted average vector of Bi-LSTM hidden vectors. This affects the model performance negatively and F1 score of the model decreases by 3.0% (A3$-$A6).

5.3 Performance with Varying Sentence Length and Varying Entity Pair Distance

We analyze the effects of our attention model with different sentence lengths in the two datasets in Figures 5 and 6. We also analyze the effects of our attention model with different distances be-

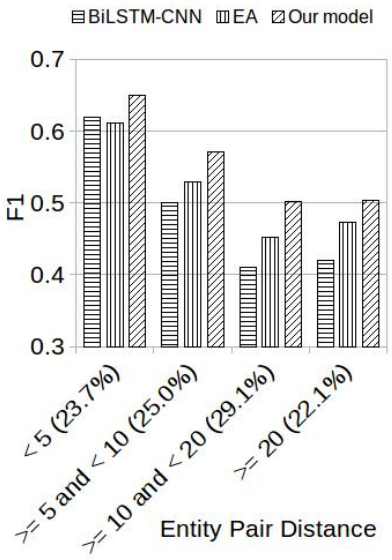


Figure 7: Performance comparison across different distances between entities on the NYT10 dataset.

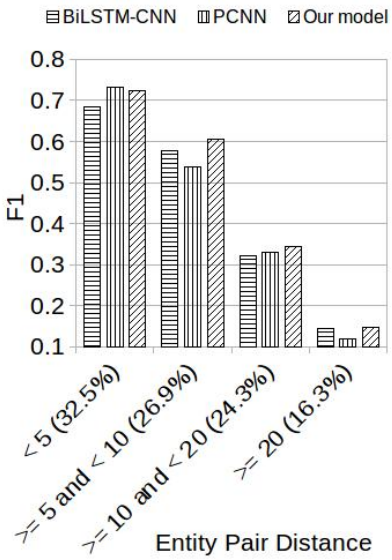


Figure 8: Performance comparison across different distances between entities on the NYT11 dataset.

	Prec.	Rec.	F1
(A1) BiLSTM-CNN	0.473	0.606	0.531
(A2) Standard attention	0.466	0.638	0.539
(A3) Window size $(ws) = 5$	0.507	0.652	**0.571**
(A4) Window size $(ws) = 10$	0.510	0.640	0.568
(A5) Softmax	0.490	0.658	0.562
(A6) Max-pool	0.492	0.600	0.541

Table 4: Effectiveness of model components ($m = 4$) on the NYT11 dataset.

tween the two entities in the two datasets in Figures 7 and 8. We observe that with increasing sentence length and increasing distance between the two entities, the performance of all models drops. This shows that finding the relation between entities located far from each other is a more difficult task. Our multi-factor attention model with dependency-distance weight factor increases the F1 score in all configurations when compared to previous state-of-the-art models on both datasets.

6 Related Work

Relation extraction from a distantly supervised dataset is an important task and many researchers (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2011) tried to solve this task using feature-based classification models. Recently, Zeng et al. (2014, 2015) used CNN models for this task which can extract features automatically. Shen and Huang (2016) and Jat et al. (2017) used attention mechanism in their model to improve performance. Surdeanu et al. (2012), Lin et al. (2016), Vashishth et al. (2018), Wu et al. (2019), and Ye and Ling (2019) used multiple sentences in a multi-instance relation extraction setting to capture the features located in multiple sentences for a pair of entities. In their evaluation setting, they evaluated model performance by considering multiple sentences having the same pair of entities as a single test instance. On the other hand, our model and the previous models that we compare to in this paper (Zeng et al., 2014, 2015; Shen and Huang, 2016; Jat et al., 2017) work on each sentence independently and are evaluated at the sentence level. Since there may not be multiple sentences that contain a pair of entities, it is important to improve the task performance at the sentence level. Future work can explore the integration of our sentence-level attention model in a multi-instance relation extraction framework.

Not much previous research has exploited the dependency structure of a sentence in different ways for relation extraction. Xu et al. (2015) and Miwa and Bansal (2016) used an LSTM network and the shortest dependency path between two entities to find the relation between them. Huang et al. (2017) used the dependency structure of a sentence for the slot-filling task which is close to the relation extraction task. Liu et al. (2015) exploited the shortest dependency path between two entities and the sub-trees attached to that path (augmented dependency path) for relation extraction. Zhang et al. (2018) and Guo et al. (2019)

used graph convolution networks with pruned dependency tree structures for this task. In this work, we have incorporated the dependency distance of the words in a sentence from the two entities in a multi-factor attention mechanism to improve sentence-level relation extraction.

Attention-based neural networks are quite successful for many other NLP tasks. Bahdanau et al. (2015) and Luong et al. (2015) used attention models for neural machine translation, Seo et al. (2017) used attention mechanism for answer span extraction. Vaswani et al. (2017) and Kundu and Ng (2018) used multi-head or multi-factor attention models for machine translation and answer span extraction respectively. He et al. (2018) used dependency distance-focused word attention model for aspect-based sentiment analysis.

7 Conclusion

In this paper, we have proposed a multi-factor attention model utilizing syntactic structure for relation extraction. The syntactic structure component of our model helps to identify important words in a sentence and the multi-factor component helps to gather different pieces of evidence present in a sentence. Together, these two components improve the performance of our model on this task, and our model outperforms previous state-of-the-art models when evaluated on the New York Times (NYT) corpus, achieving significantly higher F1 scores.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable and constructive comments on this work.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57nd Annual Meeting of the Association for Computational Linguistics*.

Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th International Conference on Computational Linguistics*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Lifu Huang, Avirup Sil, Heng Ji, and Radu Florian. 2017. Improving slot filling performance with attentive neural networks on dependency structures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2017. Improving distantly supervised relation extraction using word and entity based attention. In *Proceedings of the 6th Workshop on Automated Knowledge Base Construction*.

Souvik Kundu and Hwee Tou Ng. 2018. A question-focused multi-factor attention network for question answering. In *Proceedings of the Association for the Advancement of Artificial Intelligence*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the International Conference on Learning Representations*.

Yatian Shen and Xuanjing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of the 26th International Conference on Computational Linguistics*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems*.

Shanchan Wu, Kai Fan, and Qiong Zhang. 2019. Improving distantly supervised relation extraction with neural noise converter and conditional optimal selector. In *Proceedings of the Association for the Advancement of Artificial Intelligence*.

Yuning Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Zhi-Xiu Ye and Zhen-Hua Ling. 2019. Distant supervision relation extraction with intra-bag and inter-bag attentions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of the 25th International Conference on Computational Linguistics*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Exploiting the Entity Type Sequence to Benefit Event Detection

Yuze Ji,[1,2] **Youfang Lin,**[1,2] **Jianwei Gao,**[1,2] **Huaiyu Wan**[1,2*]
[1]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
[2]Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing, China
{yuzeji, yflin, gaojianwei, hywan}@bjtu.edu.cn

Abstract

Event Detection (ED) is one of the most important tasks in the field of information extraction. The goal of ED is to find triggers in sentences and classify them into different event types. In previous works, the information of entity types are commonly utilized to benefit event detection. However, the sequential features of entity types have not been well utilized yet in the existing ED methods. In this paper, we propose a novel ED approach which learns sequential features from word sequences and entity type sequences separately, and combines these two types of sequential features with the help of a trigger-entity interaction learning module. The experimental results demonstrate that our proposed approach outperforms the state-of-the-art methods.

1 Introduction

Event Extraction (EE) is one of the essential tasks of Information Extraction, which aims to extract structured events from unstructured texts. According to ACE (Automatic Context Extraction) event annotation guideline[1], an event is represented by an event trigger, which is often a single verb or noun, and a set of event arguments, the participants of the event. Event Detection (ED), as a crucial step in EE task, focuses on finding event trigger words and classifying them into different event types. As pointed out in (Liu et al., 2019; Ritter et al., 2012), the ambiguity in natural languages makes ED a challenging task. On the one hand, various expressions can be used to represent the same event type; on the other hand, the same event triggers, when placed in different context, can be categorized in totally different event types. To illustrate the second phenomenon, we present two examples from the widely used ACE 2005[2] dataset:

1) A Russian Soyuz capsule (VEH) **dropped** (*Transport*) the astronauts (PER) off this morning.

2) U.S. planes (VEH) **dropped** (*Attack*) a bomb (WEA) near northern Iraq.

Notice that the same annotated event trigger **dropped** has different meanings in the two sentences above and thus evokes entirely different event types. In the first sentence, **dropped** evokes a *Transport* event, but in the second sentence, **dropped** represents an *Attack* event. According to Liu et al. (2018a), in the ACE 2005 dataset, 57% of the event triggers are ambiguous. How to alleviate the ambiguity of event triggers has become a crucial problem in the ED task.

In several previous works, researchers have proved that entity mentions could play a positive role on alleviating the ambiguity of event triggers (Hong et al., 2011; Li et al., 2013; Feng et al., 2016; Liu et al., 2017, 2019, 2018a). An entity mention, as described in ACE 2005 dataset, is a reference to an object or a set of objects in the world. Back to the two sentences above, entity mentions are the underlined word tokens. In the first sentence, before classifying the event trigger **dropped**, if an event detector can obtain the information that "Soyuz capsule" is an entity mention with the type VEHICLE, and "astronauts" has the entity type PERSON, the event detector may tend to consider the potential link between VEHICLE and PERSON, which makes the event more likely to be a *Transport* event. In the second sentence, besides "planes" which is a VEHICLE type entity mention, another entity mention "bomb", which has the type WEAPON, can largely effect on the

*Corresponding author: hywan@bjtu.edu.cn
[1]https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf
[2]https://catalog.ldc.upenn.edu/LDC2006T06

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 613–623
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

ED result. The appearance of WEAPON tells the event detector that there is a weapon in the context, and this clue leads the detector to recognize an *Attack* event with less ambiguity.

Liu et al. (2017) utilize raw entity types directly as local context of words to calculate the attention value between entity types and candidate triggers, aiming to catch the most important entity type information. After that, Liu et al. (2018b, 2019); Nguyen and Grishman (2018) apply entity types as a kind of supplementary information of the word tokens. They concatenate these two types of features and feed them into neural networks to learn mixed representations.

Intuitively, we understand that the sequential features of words are very important in modeling the sentence information, since the order of tokens can largely influence the meaning of the sentence. In the previous works, different neural network architectures have been employed to capture the sequential features from word sequences (Chen et al., 2015; Lin et al., 2018; Sha et al., 2018). Similar to the word sequences, entity type sequences, which consist of entity type annotations for each token in the word sequences, also contain sequential features, because the position of an entity mention's type in the sequences may affect its importance in the ED process. However, to the best of our knowledge, there is no study which regards the entity type sequence as an independent sequence to capture the sequential features and discusses what influence the entity types' sequential features would take to the ED task.

In order to make use of the information from both entity type sequences and word sequences, in this paper, we propose a novel ED approach Entity-Type-Enhanced-Event-Detection (**ETEED**). We consider that the word sequence and the entity type sequence have equal importance and thus the representation of each sequence should be learned separately. In this procedure, the Transformer Encoder structure (Vaswani et al., 2017) is utilized to capture the sequential features. Besides, an attention based trigger-entity interaction learning module is proposed to learn the correlation between triggers and entity types. Different from previous works which calculate the attention value between a candidate trigger and the whole entity type sequence, this module only learns the relation between entity mentions' types and the candidate trigger, and returns weighted summed entity mention type representations to benefit the classification. In this way, we can avoid the disturbance from irrelevant entity types, and focus only on the effect brought by entity mentions.

In summary, our contribution in this work is as follows:

- we propose to learn entity type representations separately from word representations, in order to make full use of the sequential features from entity type sequences.
- we propose an attention-based trigger-entity interaction learning module, which focuses only on the relation between entity mentions and candidate triggers, thus can eliminate the influence brought by irrelative entity types.
- we extensively evaluate our approach on a widely used benchmark dataset ACE 2005, and the evaluation result shows that our method can achieve competitive results compared with the state-of-the-art methods.

2 Approach

In this section, we elaborate the proposed **ETEED** method. Similar to the existing works, we regard ED as a classification problem. Specifically, in each sample, there is a candidate trigger, and our goal is to classify this candidate trigger into 34 event types (33 event subtypes and a NA type). We present the overall framework of our method in Figure 1. To well illustrate our model, we divide this section into three different parts: i) token representation learning, which involves representing the word sequences and entity type sequences, ii) attention-based feature learning, which interprets how the attention module works to learn the relation between entity mentions and candidate triggers, and iii) trigger classification, which concatenates all the continuous representation together and produce the final output for the trigger classification.

2.1 Token Representation Learning

For each sample with length n in the dataset, we represent its word sequence as $w = \{w_1, w_2, .., w_n\}$, in which w_i means the i-th word in the sentence. Similarly, let $e = \{e_1, e_2, ..., e_n\}$ be the entity types

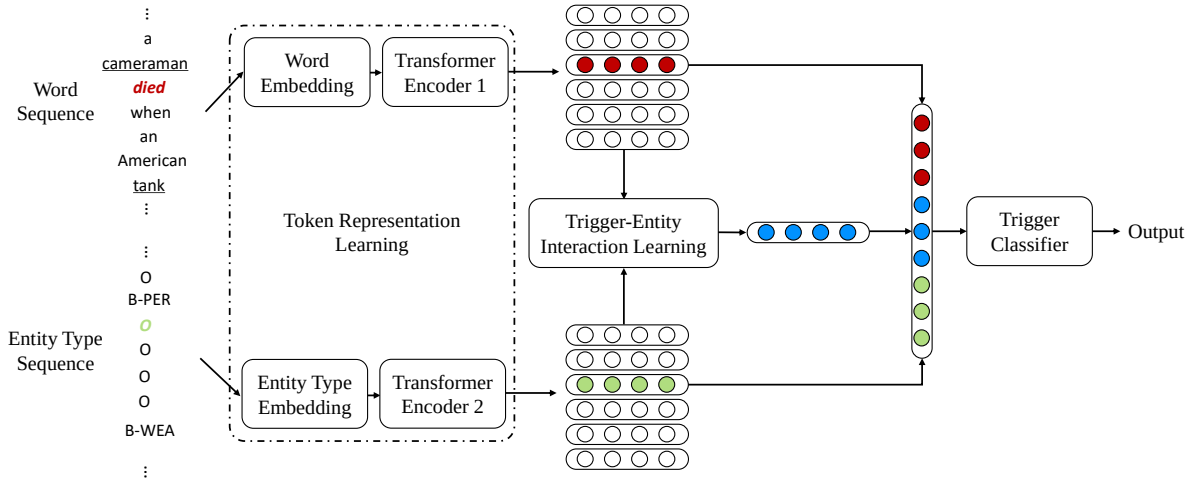


Figure 1: Global structure of our method.

corresponding to w. In consideration that cross-word entity mentions occur frequently in the ACE 2005 dataset, we apply BIO annotation schema to assign entity types for each token in the word sequence. Besides, we use c to represent the position of a candidate trigger in the sequence, so w_c and e_c show the word and entity type information of the candidate trigger respectively. In order to learn vector representations in local semantic context for the word and entity type tokens, firstly, we use embedding layers to transform the symbolic representations w and e to real-value vectors. Then, the Transformer Encoders are applied to capture the semantic relation between tokens, and learn a specific vector representation for each token.

Embedding Layer

With the help of the embedding layer, we transform the word token w_i and entity type token e_i in the input sequences into real-value representations. By looking up pre-trained word embedding matrix for w_i, a fixed sized vector representation x_{w_i} can be obtained. On the other side, for embedding entity type tokens, following existing works (Li et al., 2013; Chen et al., 2015; Liu et al., 2017; Nguyen and Grishman, 2015), we randomly initialize the real-value representation for each entity type and update it during the training process. The vector representation of e_i is marked as x_{e_i}.

Transformer Encoder Structure

Proposed by Vaswani et al. (2017), the Transformer has proved its effectiveness on the machine translation task. Different from most neural network based machine translation models (Cho et al., 2014; Bahdanau et al., 2015; Gehring et al., 2017), the Transformer is solely based on attention mechanisms, dispensing with recurrences and convolutions entirely. One of the most important reasons is that it is easier for attention mechanisms to learn long-range dependencies, which is a key challenge in sequential data modeling, than recurrent and convolutional neural networks. The Transformer has an encoder-decoder structure, the encoder maps the input sequence to new continuous representations, then the decoder receives the representations and generates an output sequence. The encoder-decoder structure is suitable for the machine translation task, however, in our approach, we only need to produce token representations for input sequences based on their local context. So, only the Transformer Encoder is used in our model.

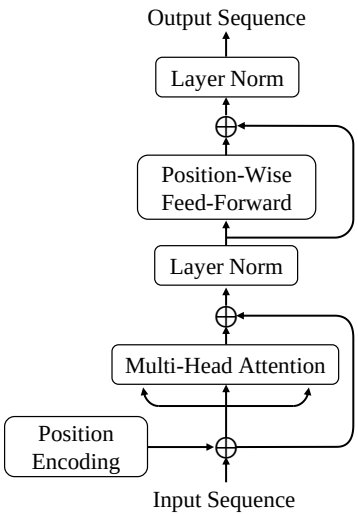


Figure 2: Architecture of the Transformer Encoder.

Figure 2 shows the architecture of the Transformer Encoder employed in our approach. For an input sequence consists of vector representations $\{x_1, x_2, ..., x_n\}$ with $x_i \in \mathbb{R}^d$, the Transformer Encoder produces the output sequence $\{z_1, z_2, ..., z_n\}$ of the same dimension with the input sequence. There are two sub-layers in the Transformer Encoder, one is a multi-head self attention layer, and the ohter is a position-wise feed-forward layer. Each sub-layer is followed by a residual connection (He et al., 2016) and a layer normalization (Lei Ba et al., 2016).

Based on single attention function, multi-head attention mechanism jointly captures the information from different representation subspaces. It firstly does h times different linear projections on the same input, then performs h single attention functions in parallel. Finally, in order to integrate all the information together, h output values from single attention functions are concatenated and projected to the same dimension with the input.

$$A_i(x) = softmax\left(\frac{\left(xW_i^Q\right)\left(xW_i^K\right)^T}{\sqrt{d_k}}\right) \quad (1)$$

$$head_i = A_i(x) \cdot \left(xW_i^V\right) \quad (2)$$

$$MultiHead(x) = [head_1, ..., head_h] \cdot W^O \quad (3)$$

where $x \in \mathbb{R}^{n\times d}$ is the input of the multi-head attention layer, $W_i^Q \in \mathbb{R}^{d\times d_k}$, $W_i^K \in \mathbb{R}^{d\times d_k}$, $W_i^V \in \mathbb{R}^{d\times d_v}$ are parameters to perform linear projections on input vectors, $W^O \in \mathbb{R}^{hd_v\times d}$ projects the concatenation of h single attention results to the same dimension with x. d_k, d_v are hyper-parameters determining projection dimensions.

In addition to the multi-head attention layer, a position-wise feed-forward layer is employed to enhance the representation capability of the Transformer Encoder.

$$FFN(x') = relu\left(x'W_1 + b_1\right)W_2 + b_2 \quad (4)$$

where $x' \in \mathbb{R}^{n\times d}$ is the input of the feed-forward layer, $W_1 \in \mathbb{R}^{d\times d_{hidden}}$, $b_1 \in \mathbb{R}^{d_{hidden}}$, $W_2 \in \mathbb{R}^{d_{hidden}\times d}$, $b_2 \in \mathbb{R}^d$, in which d_{hidden} is a hyper-parameter. Besides, since the Transformer Encoder contains neither recurrence nor convolution, in order to utilize the order information of sequence, position encodings are added to the input sequence at the beginning of the Transformer Encoder. The position encodings are calculated with the following equations:

$$PE(pos, 2i) = sin\left(pos/10000^{2i/d}\right) \quad (5)$$

$$PE(pos, 2i+1) = cos\left(pos/10000^{2i/d}\right) \quad (6)$$

where pos is the position and i is the dimension.

With the Transformer Encoder architecture described above, we obtain the vector representations from word sequence x_w and entity type sequence x_e separately, which will be marked as $z_w = \{z_{w_1}, ..., z_{w_n}\}$, $z_{w_i} \in \mathbb{R}^{d_w}$ and $z_e = \{z_{e_1}, ..., z_{e_n}\}$, $z_{e_i} \in \mathbb{R}^{d_e}$ in the following paragraph.

2.2 Trigger-Entity Interaction Learning

After the token representation learning, we get two sequences z_w and z_e. In order to encode the interaction between entity types and the candidate trigger, we introduce a trigger-entity interaction learning module in this subsection. It is an attention-based module which calculates the attention factors between the candidate trigger and entity types. In this procedure, we notice that there are many tokens with type O in the entity type sequence, which may prevent the method from explicitly modeling the relation between entity mentions' types (non-O entity types) and candidate trigger. Inspired by Nguyen and Grishman (2018), in this paper, we exclusively calculate the relation between the type of entity mentions and the candidate trigger. Given the vector representation of the candidate trigger z_{w_c} and the entity type sequence z_e, the attention values will be calculate as:

$$z_{em} = \{z_{e_i} | 1 \le i \le n \text{ and } e_i \neq O\} \quad (7)$$

$$\alpha = \sigma\left(\frac{(z_{w_c}U_1)\cdot(z_{em}U_2)^T}{\sqrt{d_w}} + b_3\right) \quad (8)$$

$$\alpha' = softmax\left(\alpha\right) \quad (9)$$

where $z_{em} \in \mathbb{R}^{k\times d_e}$ with supposing that there are k entity mentions in the sample, $U_1 \in \mathbb{R}^{d_w\times d_w}$, $U_2 \in \mathbb{R}^{d_e\times d_w}$, $b_3 \in \mathbb{R}^k$. As mentioned by Vaswani et al. (2017), to counteract the effect that the large values of d_w may impact the softmax result, we employ a coefficient $\frac{1}{\sqrt{d_w}}$ to scale the dot product.

Finally, with the help of the attention values α' and the entity mention type sequence z_{em}, we can calculate the vector representation for the trigger-entity interaction. Before doing the dot product

with α', in order to reinforce the capability of the model, we employ a fully connected layer on z_{em} with ReLU as activation function.

$$r = \alpha' \cdot relu(z_{em}U_3 + b_4) \quad (10)$$

where $U_3 \in \mathbb{R}^{d_e \times d_w}$, $b_4 \in \mathbb{R}^{d_w}$, and $r \in \mathbb{R}^{d_w}$ represents the output of trigger-entity interaction learning.

2.3 Trigger Classification

As illustrated in Figure 1, for each sample, we concatenate the trigger-entity interaction r with the candidate trigger z_{w_c} and the corresponding entity type representation z_{e_c}, to form the complete candidate trigger representation. Then, a multi-layer perceptron is employed as the trigger classifier to model the candidate trigger representation. The activation function SELU (Klambauer et al., 2017) is utilized in this multi-layer perceptron. Finally, the softmax function is applied to calculate the conditional probabilities that the candidate trigger belongs to each event type.

$$H_{c_1} = selu\left([z_{w_c}, z_{e_c}, r]W_{c_1} + b_{c_1}\right) \quad (11)$$
$$H_{c_2} = selu\left(H_{c_1}W_{c_2} + b_{c_2}\right) \quad (12)$$
$$O = softmax\left(H_{c_2}W_o + b_o\right) \quad (13)$$

where $W_{c_1} \in \mathbb{R}^{(d_e+2d_w)\times d_{c_1}}$, $W_{c_2} \in \mathbb{R}^{d_{c_1}\times d_{c_2}}$, $W_o \in \mathbb{R}^{d_{c_2}\times d_T}$ are weight metrics, in which d_{c_i} $(i \in \{1, 2\})$ are dimensions of the hidden states, d_T demonstrates the number of event types.

During the training procedure, we set the cross-entropy error as the loss function of our model, and the Adam optimizer (Kingma and Ba, 2015) is utilized to update the parameters. To keep the scale of gradients roughly the same in all layers, all the parameters are initialized by Xavier initializer (Glorot and Bengio, 2010). Table 1 shows the hyper-parameter settings in our experiments.

Module	Parameter	Value
Embedding	word	200
	entity	128
Transformer Encoder (Common)	head number	4
	layer number	1
	d_{hidden}	2048
Transformer Encoder (Word)	d_k	200
	d_v	200
Transformer Encoder (Entity)	d_k	128
	d_v	128
Trigger Classifier	d_{c_1}	256
	d_{c_2}	64
Adam Optimizer	lr	5e-5
	β_1	0.9
	β_2	0.999
	ϵ	1e-8

Table 1: Hyper-parameter settings in our experiments.

3 Experiments

3.1 Dataset

We evaluate our approach on a widely used benchmark dataset ACE 2005. In this dataset, event triggers are categorized into 33 subtypes (e.g., Be-Born, Marry, Attack). Besides, we annotate the candidate triggers which have no event types with the NA type. So, in total 34 event types are involved in our experiments. We also utilize the golden entity mentions annotated in ACE 2005 with BIO schema to produce the entity type sequences. Following the previous studies (Liu et al., 2019; Hong et al., 2018; Ji and Grishman, 2008), from the ACE 2005 English corpus, we choose randomly 40 newswire articles as the test set, 30 other articles as the development set, and pick the remaining 529 articles as the training set.

Following the ACE 2005's guideline document and Liu et al. (2019), we enumerate every noun, verb and adjective in sentences as candidate triggers. The NLP toolkit NLTK[3] is employed to parse and annotate the POS tags for sentences. We use pre-trained GloVe (Pennington et al., 2014) vectors as the embeddings for word tokens, and randomly initialize the embeddings for entity types then update them during the training procedure.

3.2 Overall Performance

We compare our ETEED model with the following state-of-the-art methods:

1) **JointBeam** is a feature-based method proposed by Li et al. (2013), which combines the manually designed local and global features to extract events.

2) **RBPB** is proposed by Sha et al. (2016), which simultaneously utilizes patterns and elaborately designed features to extract event triggers. In addition, a regularization method is applied to further improve the performance of the model.

3) **JRNN** is proposed by Nguyen et al. (2016), which combines the manually designed features with BiGRU to jointly extract triggers and arguments.

4) **HNN** is a language independent neural net-

[3]http://www.nltk.org/

Method	Trigger Identification(%)			Trigger Classification(%)		
	P	R	F1	P	R	F1
JointBeam	76.9	65.0	70.4	73.7	62.3	67.5
RBPB		N/A		70.3	67.5	68.9
JRNN	68.5	75.7	71.9	66.0	73.0	69.3
HNN*	**80.8**	71.5	75.9	**84.6**	64.9	73.4
SELF	75.3	78.8	77.0	71.3	74.7	73.0
DEEB-RNN		N/A		72.3	75.8	74.0
TEACHER		N/A		76.8	72.9	74.8
ETEED (ours)	78.1	**82.5**	**80.2**	74.1	**78.2**	**76.1**

Table 2: Overall performance with golden entity labels. * represents the model doesn't utilize entity type information.

work architecture proposed by Feng et al. (2016). With the structure which combines BiLSTM with CNN, this method can capture both the sequence and chunk information of words to benefit ED.

5) **DEEB-RNN** is proposed by Zhao et al. (2018), which incorporates the document-level clues with BiGRU to enhance ED.

6) **SELF** is proposed by Hong et al. (2018), which integrates BiLSTM into GAN structure, in order to distinguish the authentic information from spurious features.

7) **TEACHER** is an adversarial imitation based knowledge distillation approach proposed by Liu et al. (2019). This model contains two modules, one is a "teacher" module which combines the word sequences with the golden annotations of entity types and argument roles to learn knowledge representations. The other one is a "student" module which tries to imitate the representations from the "teacher" module.

We evaluate the performance via Precision (P), Recall (R) and F1-score (F1). Table 2 shows the overall performance of different approaches on the ED task. From the results, it can be found that our approach outperforms the state-of-the-art methods in both trigger identification and trigger classification. In the trigger identification, our approach achieves better results than all the previous methods in recall and F1-score (promote respectively 3.7% and 3.2% against the best baseline model SELF). Besides, although lower in precision by 2.8% , ETEED is 11% higher than the HNN model in recall. The same conclusion can be made in the trigger classification, our methods produces the highest recall and F1-score, and the improvement of F1-score is 1.3% over the best baseline model TEACHER. To summarize, on the one hand, ETEED significantly improves the recall values compared with the state-of-the-art methods. On the other hand, ETEED produces relatively comparable precisions to the existing methods, which ensures the good F1-score.

Method	Classification F1(%)
JointBeam	65.6 (↓1.9)
RBPB	67.8 (↓1.1)
TEACHER	71.2 (↓3.6)
ETEED (ours)	**74.8** (↓1.3)

Table 3: Performance with predicted entity labels. ↓ represents the performance drop from golden annotations.

To further verify the performance of our model in the real testing scenario, where the golden entity annotations are missing, we utilize the predicted entity type labels in the test procedure. Following Liu et al. (2019), we train a BiLSTM-CRF model on the training set, then apply it on the test set to get the predicted entity type sequences. The F1-score of the BiLSTM-CRF model on the test set is 82.7%. The JointBeam, RBPB and TEACHER are selected as baseline methods. Table 3 shows that our approach has significant improvement compared with the baseline methods. Besides, our model has relatively small performance descent with using predicted annotations than using golden annotations.

3.3 Effect of Entity Type Representation

In order to evaluate the effect of the entity type representation, we design the two following experiments:

Method	Trigger Identification(%)			Trigger Classification(%)		
	P	R	F1	P	R	F1
ETEED$_{no_entity}$	82.1	74.4	78.0	77.8	70.5	74.0
ETEED$_{concat}$	**83.2**	73.9	78.3	**78.4**	69.7	73.8
ETEED	78.1	**82.5**	**80.2**	74.1	**78.2**	**76.1**

Table 4: Effect of entity type sequence representation.

Method	Trigger Identification(%)			Trigger Classification(%)		
	P	R	F1	P	R	F1
ETEED$_{no_interaction}$	**82.6**	75.2	78.7	**79.3**	72.2	75.6
ETEED$_{all_entity}$	82.3	77.4	79.7	78.2	73.5	75.8
ETEED	78.1	**82.5**	**80.2**	74.1	**78.2**	**76.1**
TEACHER	N/A			76.8	72.9	74.8
ETEED$_{argument}$	87.4	82.9	85.1	**86.0**	**81.6**	**83.8**

Table 5: Effect of trigger-entity interaction.

1) **ETEED**$_{no_entity}$ utilizes only word token sequences, which means there is no more entity information used in the model. For each sample, firstly the word sequence is encoded by the Transformer Encoder, then the candidate trigger's representation x_{w_c} is fed into the trigger classifier to get its event type.

2) **ETEED**$_{concat}$ follows some previous works (Liu et al., 2019, 2017), this model utilizes the concatenation of the word representations and the entity type representations as the input of downstream structures. In ETEED$_{concat}$, the concatenated representations are sent into the Transformer Encoder, then the features of the candidate triggers are picked from the output results to be classified by the trigger classifier.

The hyper-parameter settings keep the same with the ETEED. Table 4 shows that, our approach significantly outperforms the baseline models on F1-score (1.9% on identification and 2.1% on classification). In the meanwhile, our method has higher recall values but slightly lower precision values than the baseline models. This phenomenon shows that our method can largely improve the recall values with few precision loss. Especially, when compared with ETEED$_{concat}$, ETEED produces large performance gain on F1-score (1.9% and 2.1% on identification and classification respectively), which proves the effectiveness of our method. With the entity types' sequential features, our method can bring more entity information to trigger classifier than the baseline models.

3.4 Effect of Trigger-Entity Interaction

In this subsection, we conduct three comparison experiments to evaluate the effect of the trigger-entity interaction learning.

1) **ETEED**$_{no_interaction}$ removes the trigger-entity learning from the complete model.

2) **ETEED**$_{all_entity}$ uses all entity type information rather than the entity mentions' type information to learn trigger-entity interaction.

3) **ETEED**$_{argument}$ is a model designed to evaluate the effect of the argument role sequence. As mentioned in (Liu et al., 2019), TEACHER needs to utilize both the entity type and the argument role information to get the best performance. In the ETEED$_{argument}$, we replace the entity type sequences by the argument role sequences to compare with the TEACHER model.

Table 5 shows the results. Compared with ETEED$_{no_interaction}$, the complete ETEED model performs larger improvement on identification than classification on F1-score (1.5% to 0.5%). This result reveals that the use of entity mentions' types is obviously helpful when judging the occurrence of an event, but when it comes to trigger classification, the entity mention information brings less improvement. In order to explicitly capture the entity mention information, we only use entity mention features in our approach. To evaluate its effect, we compare ETEED$_{all_entity}$ with ETEED. Although the performance gain are

not significant, focusing on entity mentions can still bring 0.5% and 0.3% F1-score improvement on trigger identification and classification.

When comparing ETEED$_{argument}$ with ETEED, we can find that the use of argument role information can dramatically improve the performance of ED, because argument roles contain more information than entity types. Besides, some special argument roles can point to specific event types. As methods which use argument role information, ETEED$_{argument}$ significantly outperforms TEACHER, this result further proves the effectiveness of our model. However, in the real testing and application scenarios, we can hardly obtain the arguments role information before getting the event types of the candidate triggers. Instead, using predicted entity type information is more feasible.

4 Related Work

Event Detection is an important task in Information Extraction. The majority of existing approaches regard this task as a classification problem, and we summarize these approaches into two categories globally.

Feature-based methods are proposed as the first kind of approach to tackle the ED task by introducing feature-engineering to convert the classification clues like POS tags and dependency features into feature vectors (Ahn, 2006; Ji and Grishman, 2008; Hong et al., 2011; Li et al., 2013; Patwardhan and Riloff, 2009; Gupta and Ji, 2009; Liao and Grishman, 2010; Liu et al., 2016). This kind of approach depends heavily on expert knowledge and manual feature design, which makes these approaches time-consuming and low adaptability on different datasets. Chambers and Jurafsky (2011) design a weakly supervised system, which can automatically induce the event templates and extract event information from unlabeled corpus, to alleviate the need of expert knowledge. However, the required external resources are not always available for some low-resource languages.

In recent years, deep learning methods have proved their effectiveness on the ED task. (Chen et al., 2015; Nguyen and Grishman, 2016; Lin et al., 2018) utilize CNN to automatically capture the high-level vector representations of sentences. Nguyen et al. (2016) apply RNN in their model in order to capture the sequential features in the sentences. Feng et al. (2016) combine CNN with RNN and propose a hybrid neural network. Araki and Mitamura (2018) make use of the distant supervision mechanism to detect the events regardless of domains. Liu et al. (2017); Zhao et al. (2018) utilize attention mechanisms aiming to fuse the external sentence features (i.e. entity type features, document features) with the word features. (Hong et al., 2018; Liu et al., 2019) implement adversarial training to distinguish effective information from spurious features. GCN is a powerful neural network architecture on graphs, Nguyen and Grishman (2018) use this architecture to represent dependency relations in sentences. Compared with the feature-based methods, the deep learning methods need no more feature-engineering, which means less financial/time cost and better adaptability. Among them, there are several approaches which utilize the entity type information in their neural networks. Liu et al. (2017) utilize the entity type embedding directly as local context of the current word, and calculate the attention values between them; others (Liu et al., 2018b, 2019; Nguyen and Grishman, 2018) concatenate the entity type embeddings with the work token embeddings, in order to integrate these two types of features into mixed representations with the help of neural networks. However, these existing studies ignore the entity types' sequential features which may benefit the ED task.

In our approach, we learn the word features and the entity type features separately, which allows us to capture the sequential features of entity types and thus make full use of the entity type information. Besides, an attention-based trigger-entity interaction learning is introduced in our work to learn relations between the candidate trigger words and the entity mentions' type features.

5 Conclusion

In this work, we propose a novel neural network architecture ETEED for the ED task. In order to capture the sequential features from both the word sequences and the entity type sequences, our approach proposes to model these two types of sequences separately. The two types of sequences are firstly modeled by two isolated Transformer Encoders, then, an attention-based trigger-entity interaction learning module is applied to capture the correlations between the candidate trigger's word representation and the entity type representations of the sequence. Aiming to obtain a more

accurate interaction representation, this module utilizes only the entity mentions' type information rather than the whole entity type sequence to calculate the attention values. Finally, the concatenation of the candidate trigger's word, entity type representations and the trigger-entity interaction representation is fed into the trigger classifier to obtain the final event category. In the future, we plan to extend our method to the Event Extraction task, which means not only to extract triggers from sentences, but also to identify and classify the corresponding event arguments.

Acknowledgments

This paper is supported by the National Key R&D Program of China (No. 2018YFC0830200). We thank anonymous reviewers for their valuable comments.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.

Jun Araki and Teruko Mitamura. 2018. Open-domain event detection using distant supervision. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 878–891, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 976–986, Portland, Oregon, USA. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 66–71, Berlin, Germany. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 369–372, Suntec, Singapore. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1127–1136, Portland, Oregon, USA. Association for Computational Linguistics.

Yu Hong, Wenxuan Zhou, Jingli Zhang, Guodong Zhou, and Qiaoming Zhu. 2018. Self-regulation: Employing a generative adversarial network to improve event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 515–526, Melbourne, Australia. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *Computing Research Repository*, arXiv:1607.06450. Version 1.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797, Uppsala, Sweden. Association for Computational Linguistics.

Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2018. Nugget proposal networks for Chinese event detection. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1565–1574, Melbourne, Australia. Association for Computational Linguistics.

Jian Liu, Yubo Chen, and Kang Liu. 2019. Exploiting the ground-truth: An adversarial imitation based knowledge distillation approach for event detection. In *AAAI*, Honolulu, Hawaii, USA.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018a. Event detection via gated multilingual attention mechanism. In *Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Vancouver, Canada. Association for Computational Linguistics.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Thirtieth AAAI Conference on Artificial Intelligence*, pages 2993–2999. AAAI Press.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018b. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371, Beijing, China. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 886–891, Austin, Texas. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5900–5907. AAAI Press.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 151–160, Singapore. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. RBPB: Regularization-based pattern balancing method for event extraction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1224–1234, Berlin, Germany. Association for Computational Linguistics.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge RNN and tensor-based argument interaction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5916–5923.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2018. Document embedding enhanced event detection with hierarchical and supervised attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 414–419, Melbourne, Australia. Association for Computational Linguistics.

Named Entity Recognition - Is there a glass ceiling?

Tomasz Stanislawek[†,‡], **Anna Wróblewska**[†,‡], **Alicja Wójcika**[†,§],
Daniel Ziembicki[†,§] **Przemyslaw Biecek**[‡,¶]

[†]Applica.ai, Warsaw, Poland
[¶]Samsung Research Poland, Warsaw, Poland
[‡]Faculty of Mathematics and Information Science, Warsaw University of Technology
[§]Department of Formal Linguistics, University of Warsaw

Abstract

Recent developments in Named Entity Recognition (NER) have resulted in better and better models. However, is there a glass ceiling? Do we know which types of errors are still hard or even impossible to correct? In this paper, we present a detailed analysis of the types of errors in state-of-the-art machine learning (ML) methods. Our study reveals the weak and strong points of the Stanford, CMU, FLAIR, ELMO and BERT models, as well as their shared limitations. We also introduce new techniques for improving annotation, for training processes and for checking a model's quality and stability.

Presented results are based on the CoNLL 2003 data set for the English language. A new enriched semantic annotation of errors for this data set and new diagnostic data sets are attached in the supplementary materials.

1 Introduction

The problem of Named Entity Recognition (NER) was defined over 20 years ago at the Message Understanding Conference (MUC, 1995; Sundheim, 1995). Nowadays, there are a lot of solutions capable of a very high accuracy even on very hard and multi-domain data sets (Yadav and Bethard, 2018; Li et al., 2018).

Many of these solutions benefit from large available data sets or from recent developments in deep neural networks. However, in order to progress further with this last mile, we need a better understanding of the sources of errors in NER problem; as it is stated that *"The first step to address any problem is to understand it"*. We performed a detailed analysis of errors on the popular CoNLL 2003 data set (Tjong Kim Sang and De Meulder, 2003).

Of course, different models make different mistakes. Here, we have focused on models that constitute a kind of breakthrough in the NER domain. These models are: Stanford NER (Finkel et al., 2005), the model made by the NLP team from Carnegie Mellon University (CMU) (Lample et al., 2016), ELMO (Peters et al., 2018), FLAIR (Akbik et al., 2018) and BERT-Base (Devlin et al., 2018). In the Stanford model, Conditional Random Fields (CRF) with manually created features were tackled. Lample and the team (at CMU) used an LSTM deep neural network with an output with CRF for the first time. ELMO and FLAIR are new language modeling techniques as an encoder, and LSTM with a CRF layer as an output decoder. A team from Google used a fine-tuning approach with the BERT model in a NER problem for the first time, based on a Bi-diREctional Transformer language model (LM).

We analyzed the data set from a linguistic point of view in order to understand problems at a deeper level. As far as we know only a few studies analyse in details errors for NER problems (Niklaus et al., 2018; Abudukelimu et al., 2018; Ichihara et al., 2015). They mainly explore a range of name entities (boundaries in a text) and the precision and popular metrics of a class prediction (precision, recall, F1). We found the following discussions valuable:

- (Abudukelimu et al., 2018) on annotation and extraction of Named Entities,
- (Braşoveanu et al., 2018) on an analysis of errors in Named Entity Linking systems,
- (Manning, 2011) on linguistic limitations in building a perfect Part-of-Speech Tagger.

We took a different approach. First, our team of data scientists and linguists defined 4 major and

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 624–633
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

11 minor categories of types of problems typical for NLP (see Tab. 2). Next, we acquired all erroneous samples (containing errors in model outputs) and we assigned them to the newly defined categories. Finally, we characterized the incorrect output of the models with regard to gold standard annotations and following our team's consensus.

Accordingly, our overall contribution is a conceptualization and classification of the roots of problems with NER models as well as their characterization. Moreover, we have prepared new diagnostic sets for some of our categories so that other researchers can check the weakest points of their NER models.

In the following sections, we introduce our approach regarding the re-annotation process and model evaluation (section 2); we also show and discuss the results (section 3). Finally, we conclude our paper with a discussion (section 4) and draw conclusions (section 5).

2 Method

We commenced our research by reproducing the selected models for the CoNLL 2003 data set[1]. Then, we analysed the erroneous samples, sentences from the test set. It is worth mentioning that we analysed the most common types of named entities, i.e. PER - names of persons, LOC - location names, ORG - organization names. Having several times reviewed the model results and the error-prone data set, we defined the linguistic categories that are the most probable sources of model mistakes. As a result, we were able to annotate the samples with these categories; we then analysed the results and found a few possible improvements.

2.1 Models description

A brief history of the key developments of NER models for the CoNLL data is listed in Table 1. In our analysis, we chose 5 models (bold in the table) that make up significant progress.

Stanford NER CRF was the first industry-wide library to recognize NERs (Finkel et al., 2005). The LSTM layer put forward by Lample from Carnegie Mellon University (CMU) was the first deep learning architecture with a CRF output layer (Lample et al., 2016). The following: a token-based language model (LM) with bi-LSTM with CRF (ELMO) (Peters et al., 2018), a character-based LM with the same output (FLAIR) (Akbik et al., 2018) and a bi-directional language model based on an encoder block from the transformer architecture (BERT) with a fine tune classification output layer (Devlin et al., 2018) are very important techniques; and that not only in the domain of NER.

Model	F1
Ensemble of HMM, TBL, MaxEnt, RRM (Florian et al., 2003)	88.76
Semi-supervised learning (Ando and Zhang, 2005)	89.31
Stanford CRF (Finkel et al., 2005)	87.94
Neural network (Collobert et al., 2011)	89.59
CRF & lexicon embeddings (Passos et al., 2014)	90.90
CMU LSTM-CRF (Lample et al., 2016)	90.94
Bi-LSTM-CNNs-CRF (Ma and Hovy, 2016)	91.21
ELMO: Token based LM Bi-LSTM-CRF (Peters et al., 2018)	92.22
BERT-base: Fine tune Bi-Transformer LM with BPE token encoding (Devlin et al., 2018)	92.4 (*)
CVT: Cross-view training with Bi-LSTM-CRF (Clark et al., 2018)	92.61
BERT-large: Fine tune Bi-Transformer LM with BPE token encoding (Devlin et al., 2018)	92.8 (*)
FLAIR: Char based LM + Glove with Bi-LSTM-CRF (Akbik et al., 2018)	93.09 (**)
Fine tune Bi-Transformer LM with CNN token encoding (Baevski et al., 2019)	93.5

Table 1: Results reported in authors' publications about NER models on the original CoNLL 2003 test set. (*) There is no script for replicating these results and also hyper-parameters were not given. See a discussion at (google bert, 2019) (**) This result was not achieved with the current version of the library. See a discussion at (Flair, 2018) and the reported results at (Akbik et al., 2019)

2.2 Linguistic categories

From a human perspective, the task of NER involves several sources of knowledge: the situation in which the utterance was made, the context of

[1] The details of the model parameters are described in our supplementary materials.

other texts and utterances in the particular domain, the structure of the sentence, the meaning of the sentence, and general knowledge about the world.

While designing categories for annotation, we tried to define these layers of NEs understanding; however, some of them are particularly problematic. For example, there is a problem with a distinction between the meaning (of lexical items and of a whole sentence) and general knowledge. Since there is an enormous and relentless linguistic and philosophical debate on this topic (Rey, 2018), we decided not to delimit these categories and not to distinguish them. Therefore, they have been labeled together as 'sentence level context' (SL-C).

Consequently, we ended up with a set of categories for annotating the items (sentences) from our data set, which are presented in Table 2 as well as described briefly in the following sections and more precisely in the supplementary materials. We have also added more examples for each category in this material.

shortcut	linguistic property
DE-	Data set Errors
DE-A	Annotation errors
DE-WT	Word Typos
DE-BS	Word/Sentence Bad Segmentation
SL-	Sentence Level dependency
SL-S	Sentence Level Structure
SL-C	Sentence Level Context
DL-	Document Level dependency
DL-CR	Document Co-Reference
DL-S	Document Structure
DL-C	Document Context
G-	General properties
G-A	General Ambiguity
G-HC	General Hard Case
G-I	General Inconsistency

Table 2: Linguistic categories prepared for our annotation procedure.

DE-A: Annotation errors are obvious errors in the preliminary annotations (the gold standard in the CoNLL test data set). For example: in the sentence *"SOCCER - JAPAN GET LUCKY WIN, CHINA IN SURPRISE DEFEAT"* as a gold standard annotation *"CHINA"* is assigned a person type; it should, however, be defined as a location so as to be consistent with the other sentence annotations.

DE-WT: Word typos are simple typos in any word in a sample sentence, for exmple: *"Pollish"* instead of *"Polish"*.

DE-BS: Word-sentence bad segmentation. We annotated this case if a few words, joined together with a hyphen or separated by a space, were incorrectly divided into tokens (e.g. *"India-South"*), or where a sentence was erroneously divided inside a boundary of a named entity, which prevented its correct interpretation. For example: in the data set there is a sentence divided into two parts: *"Results of National Hockey"* and *"League"*.

SL-S: Sentence level structure dependency occurs when there is a special construction within a sentence (a syntactic linguistic property) that is a strong premise for defining an entity. In the studied material, we distinguished two such constructions: brackets and bullets. The error receives the SL-S annotation, when the system should have been able to recognize a syntactic linguistic property that leads to correct NER tagging but failed to do so and made a NER mistake. For example: one of the analysed NER systems did recognize all locations except *"Philippines"* in the following enumerating sentence: *"ASEAN groups Brunei, Indonesia, Malaysia, the Philippines, Singapore, Thailand and Vietnam."*.

SL-C: Sentence level context cases are those in which one is able to define an appropriate category of NE based only on the sentence context. For example: one of NER systems has a problem with recognizing the organization *"Office of Fair Trading"* in the sentence: *"Lang said he supported conditions proposed by Britain's Office of Fair Trading, which was asked to examine the case last month."*.

DL-CR: Document level co-reference category was annotated if there was a reference within a sentence to an object that was also referred to in another sentence in the same document. For example: evaluating the *"Zywiec"* named entity in the sentence *"Van Boxmeer said Zywiec had its eye on Okocim ..."*, it has to be considered that there is another sentence in the same document in the data set that explains the organization name, which is: *"Polish brewer Zywiec's 1996 profit..."*.

DL-S: Document level structure cases are those in which the structure of a document plays an important role, i.e. the occurrence of objects in the table (for example the headings determine

the scope of an entity itself and its category). For example: look at the following three sentences, which obviously compose a table: *"Port Loading Waiting"*; *"Vancouver 5 7"*, *"Prince Rupert 1 3"*. One of our NER systems had a problem with recognizing each localisation inside the table; however, the system recognized the header as a named entity.

DL-C: Document level context is a type of a linguistic category in which the entire context of a document (containing an annotated sentence) is needed in order to determine a category of an analysed entity, and in which none of the sentence level linguistic categories has been assigned (neither SL-S and SL-C).

G-A: General ambiguity are those situations in which an entity occurs in a different sense from that in which this word (entity) is used in its most common understanding and usage. For example: the common word *'pace'* may as well be occur to be a surname, as in the following sentence: *"Pace, a junior, helped Ohio State..."*.

G-HC: General hard cases are cases occurring for the first time in a set in a given subtype, and which can be interpreted in two different ways. For example: *"Real Madrid's Balkan strike force..."* where the word *'Balkan'* can be a localisation or an adjective.

G-I: General inconsistency are cases of inconsistencies in the annotation (in the test set itself as well as between the training and test sets). For example in the sentence: *"... Finance Minister Eduardo Aninat said."*, the word *'Finance'* is annotated as an organisation but in the whole data set the names of ministries are not annotated in the context of the role of a person.

2.3 Annotation procedure

All those entities that had been incorrectly recognized by any of the tested modelsfalse positives, false negatives and wrongly tagged entities were annotated in our research by two teams. Each team consisted of a linguist and a data scientist. We did not analyse errors with the MISC entity type, but the person, localisation and organisation names. The MISC type comprises a variety of NERs that are not of other types. Its definition is rather vague and it is hard to conceptualize what it actually means, e.g. if whether it comprises events or proper names, or even adjectives.

The annotation process was performed in four steps:

1. a set of linguistic annotation categories was established, see the previous section 2.2;

2. the data set was split into two equal parts: one part for each team; all entities were annotated twice, by a linguist and by a data scientist, each working independently;

3. the annotations were compared and all inconsistencies were solved within each team;

4. two teams checked the consistency of the other team's annotations; all borderline and dubious cases were discussed by all team members and reconciled.

The inter-annotator agreement statistics and Kappa are presented in Table 3. A few categories were very difficult to conceptualize, so it took more time to solve these inconsistencies. In these inconsistent cases, two annotators (a linguist and a data scientist) thoroughly discussed each example.

Not all categories (see Table 2) were annotated by the whole team. Those easy to annotate, as the categories regarding simple errors (i.e. DE-A, DE-WT, DE-BS), were done by one person and then just checked by another.

The general inconsistencies category (G-I) were done semi-automatically and then checked. The semi-automatic procedure was as follows: first finding similarly named entities in the training and test sets and then looking at their labels. By 'similarly named entities' we mean, e.g. a division of an organization having a geographical location in its name ("Pacific Division"), or a designation of a person from any country ("Czech ambassador").

Additionally, a document level context (DL-C) category was derived from the rule of not being present in any sentence level category (i.e. SL-C or SL-S).

2.4 Our diagnostic procedure

The next step, after the analysis of linguistic categories of errors, was to create additional diagnostic sets. The goal of this approach was to find, or create, more examples that reflect the most challenging linguistic properties; these can be sentence and document level dependencies and can also include a few ambiguous examples. These ambiguities are for instance names that contain words in common usage. We selected 65 examples

annotated class	agreement [%]	Kappa
SL-S	94.99	0.572
SL-C	69.64	0.389
DL-CR	78.00	0.554
DL-S	81.44	0.536
G-A	68.96	0.252
G-HC	74.46	0.340

Table 3: Inter-annotator statistics (agreement and Kappa) at the very first stage of the annotation procedure, before discussing each controversial example and the super-annotation stage. The statistics are calculated for those categories that were annotated by human annotators.

from Wikipedia articles per two groups of linguistic problems: sentence-level and document-level contexts.[2]

The first diagnostic set comprises sentences in which the properties of a language, general knowledge or a sentence structure are sufficient to identify a NE class. We use this Template Sentences (TS) to check whether a model will have the same quality after changing words, i.e. a name of an entity. For each sentence we prepared at least 2 extra entities with different lengths of words which are well suited to the context. For example in a sentence: *"Atlético's best years coincided with dominant Real Madrid teams."*, the football team *"Atlético"* can be replaced with *"Deportivo La Coruña"*.

The second batch of documents was a group of sentences in which a sentence context is not sufficient to designate a NE, so we need to know more about the particular NE, e.g. we need to look for its co-references in the document, or we require more context, e.g. a whole table of sports results, not only one row. (This particular case often occurs in the CoNLL 2003 set when referring to sports results.) We called this data set Document Context Sentences (DCS). In this data set we annotated NEs and their co-references that are also NEs. An example of such a sentence and its context is as follows: *"In 2003, Loyola Academy (X, ORG) opened a new 60-acre campus ... The property, once part of the decommissioned NAS Glenview, was purchased by Loyola (X,ORG) in 2001."* The second occurrence of the *"Loyola"* name is difficult to recognize as an organization without its first occurrence, i.e. *"Loyola Academy"*.

[2]Our prepared diagnostic data sets are available at https://github.com/applicaai/ner-resources

The other type of a diagnostic set is fairly simple. It is generated from random words and letters that are capitalized or not. Its purpose is just to check if a model over-fits a particular data set (in our case, the CoNLL 2003 set). A scrutinized model should not return any entities on those Random Sentences (RS). We generated 2 thousands of these pseudo-sentences.

3 Results

3.1 Annotation quality

In Table 4 we gathered our model's results for the standard CoNLL 2003 test set and the same set after the re-annotation and correction of annotation errors. We replaced only those annotations (gold standard) which we (all team members) were sure of. Those sentences in which the class of an entity occurrence was ambiguous were not corrected. This shows that the models are better than we thought they were, and so we corrected only the test set and left the inconsistencies.[3].

	Stanford	CMU	ELMO	FLAIR	BERT
ALL-O	88.13	89.78	92.39	92.83	91.62
ALL-C	88.73	90.39	93.21	**93.79**	92.33
PER-O	93.31	95.74	97.07	97.49	96.14
PER-C	93.94	96.49	97.81	**98.08**	96.88
ORG-O	84.23	86.90	90.68	91.34	90.61
ORG-C	84.89	87.53	91.61	**92.64**	91.44
LOC-O	90.83	92.02	93.87	94.01	92.85
LOC-C	91.58	92.62	**94.92**	94.72	93.59
MISC-O	79.10	77.31	82.31	82.89	80.81
MISC-C	79.37	77.58	82.47	**84.40**	81.10

Table 4: Results for selected models on the original (designated as ending '...-O') and re-annotated / corrected ('...-C') CoNLL 2003 test set concerning NE classes (ALL comprise PER, ORG, LOC, MISC). The given metric is a multilabel-F1 score (percentages).

3.2 Linguistic categories statistics

In the CoNLL 2003 test set, we chose as samples words and sentences in which at least one model made a mistake. The set of errors comprises 1101

[3]A small part of the data set of annotation corrections and also the debatable cases will be available at our github – https://github.com/applicaai/ner-resources. We decided not to open the whole data set, because it is the test set and the tuning models on this set would lead to unfair results. On the other hand, we could not perform the analysis on a validation set because it is rather poor with respect to different kinds of linguistic properties.

named entities. The results of each model on this set in terms of our linguistic categories are presented in Fig. 1, Fig. 2 and in Table 5.

Most mistakes were made by the Stanford and CMU models, 703 and 554 respectively. ELMO, FLAIR and BERT, which use contextualised language models, performed much better. These embedded features help the models to understand words in their context and thus resolve most problems with ambiguities.

The CMU model has most problems with sentence level context and ambiguity. This is probably due to the fact that this model uses non-contextualized embedded features (Fig. 2). The Stanford model fares the worst in terms of structured data (almost twice as many errors as the other models), which means that it is not good at defining an entity type within a very limited context (Tab. 5). The Stanford model's hand-crafted features do not store information about the probabilities of words which could represent a specific entity type. It generates much more errors than the other models.

	Stanford	CMU	ELMO	FLAIR	BERT
DE-WT	10	6	9	8	10
DE-BS	38	39	33	33	40
SL-S	46	21	13	16	11
SL-C	448	378	250	223	300
DL-CR	372	316	198	184	263
DL-S	202	107	97	100	117
DL-C	247	175	144	146	170
G-A	219	183	98	101	94
G-HC	72	68	65	59	65
G-I	19	20	21	20	20
Errors	703	554	395	370	472
Unique errors	235	93	23	12	79

Table 5: Number of errors for a particular model and a particular class of errors. The total number of annotated errors is 1101.

Modern techniques using contextualized language models like ELMO, FLAIR and BERT reduced a number of mistakes in SL-C category by more than 50% in comparison to the Stanford model. But they are unable to fix most errors in general problems related to inconsistency (G-I), general hard cases (G-HC) or word typos (DE-WT). See Figure 4 for more details.

Nevertheless, there are still a lot of common

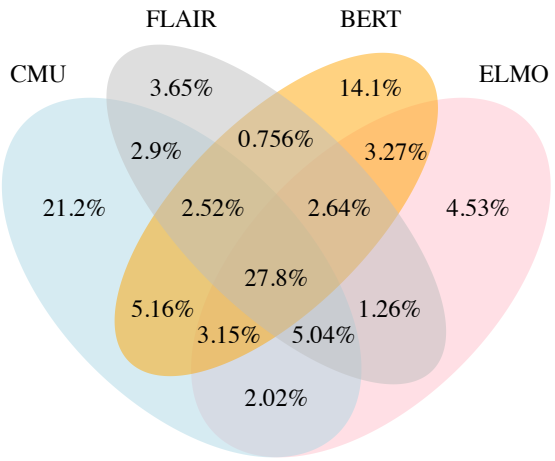


Figure 1: Venn diagram for errors in the CMU, FLAIR, BERT, ELMO models. The four models generate 794 errors and 221 are common to all of them. The Stanford model as the most error-prone is here not referred to.

problems (27.8%). In common errors (Fig. 3), SL-C (sentence level context) and DL-CR (document level co-reference) co-occur the most often. Thus, if a model also takes into account the context of a whole document, it can be of great benefit. Considering a document structure (DL-S) in modeling is also very important. This also can help to resolve a lot of ambiguity issues (G-A). Here is an example of such a situation: *"Pace outdistanced three senior finalists..."*, *"Pace"* is a person's surname, but one is able to find it out only when analysing the whole document and finding references to it in other sentences that directly point to the class of the named entity.

We must be aware of the fact that some problems cannot be resolved with this data set, not even in general. Those problems have roots in two main areas: data set annotation (word typos, bad segmentation, inconsistencies) and a complicated structure of a language. Generally in most languages it is easier to say what entity represents a real word instance than to define an exact entity type (especially when we use a metonymic sense of a word), e.g. 'Japan' can be a name of a country or of a sports team.

3.3 Diagnostic data sets

Looking at the models' results in our diagnostic data sets (Tab. 6), the first and most important observation is that we achieved significantly lower results than originally on the CoNLL 2003 test

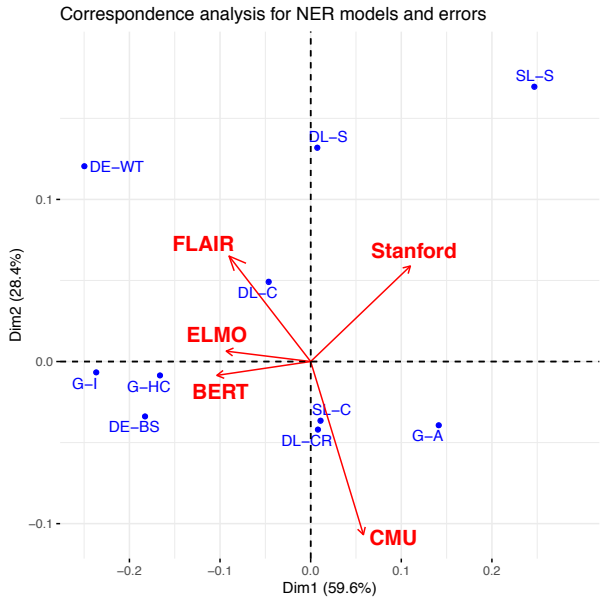


Figure 2: Correspondence analysis for the models' errors. ELMO, FLAIR and BERT are more affected by G-HC and G-I, FLAIR is also reduced with DL-C and DE-WT. See Table 5 for more details and Table 2 for names of categories.

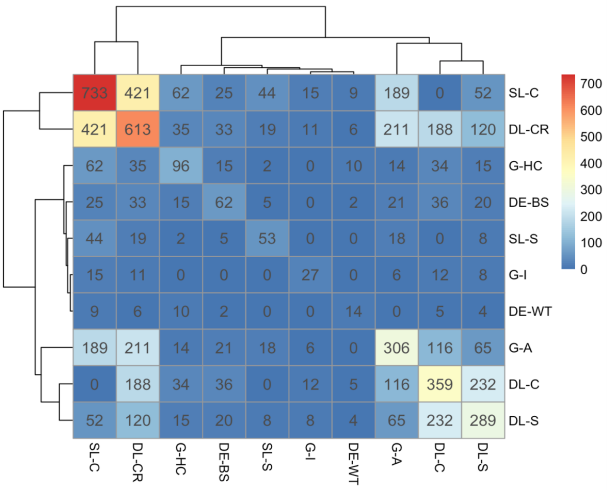


Figure 3: Heatmap for errors from the five considered models. 197 errors are common to all the models. In this figure we can see which linguistic categories tend to occur together.

set[4]. The reason for this is that diagnostic examples were selected for a broader range of topics (not only politics or sports). In particular, document context sentences (DCS) contain 364 unique entities of which only 47 appeared in an exact word form in the training data, and only 42 of them have the same entity type (organization, location or person) - the same type as in the CoNLL 2003

[4]We add statistics and a few examples from our diagnostic data sets in the supplementary materials.

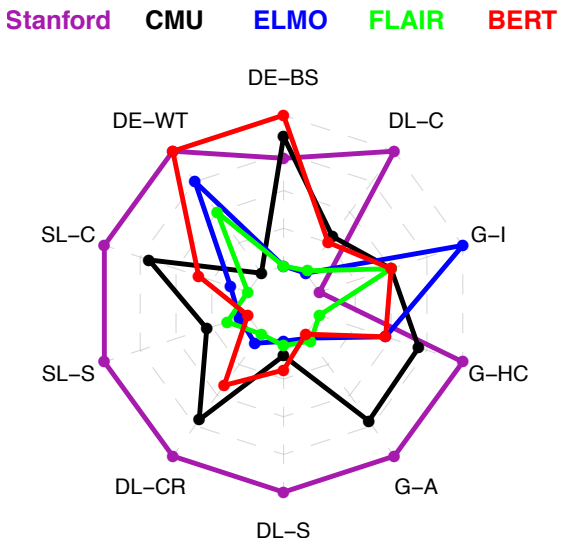


Figure 4: Radar plot with the strong and weak sides of NER models. A radius corresponds to a number of errors in a given linguistic category, the smaller the better. See Table 5 for more details.

training set. Additionally, those sentences are also difficult due to their linguistic properties (for some entities you must analyse a whole article to properly distinguish their type).

As far as the results of the diagnostic sets are concerned, we observed much better results for solutions using embeddings generated by the language models. It seems that by using ELMO embeddings we can outperform the FLAIR and BERT-Base models in case of sentences about general topics, in which the context of a whole sentence is more important than properties of words composing entities.

Moreover, when we tested all the models on random sentences (RS), this was not so good as we might have expected. All the models are very sensitive to words starting with or consisting of capital letters. Results from this diagnostic set could help to choose a model that must work properly on documents which were produced by the OCR engine with their many mistakes and misspellings.

Another interesting idea is to train or just test a model on some template sentences (TS). With such a data set we can test a model's ability to detect proper boundaries of an entity. We can do it by replacing a template entity with another one consisting of a different number of words. We could also adjust our models to a particular domain, e.g.to change entities with a PERSON type in an original data set to be more globally diversified, if we have to extract person names from the

whole world (Asian or Russian names).

	Stan-ford	CMU	ELMO	FLAIR	BERT
DCS (F1)	45.37	61.86	**76.36**	71.89	68.90
DCS (P)	43.66	58.07	73.11	69.35	59.06
DCS (R)	47.21	66.17	79.92	74.63	82.66
TS-O (F1)	68.96	79.66	**89.45**	88.51	83.47
TS-O (P)	76.92	78.33	85.48	85.25	75.18
TS-O (R)	62.50	81.03	93.81	92.04	93.81
TS-R (F1)	63.06	72.86	85.01	**86.63**	79.66
TS-R (P)	65.47	70.65	81.45	83.70	71.60
TS-R (R)	60.83	75.21	88.91	89.77	89.77
RS (No)	3571	3339	2096	**1404**	3086

Table 6: Diagnostic data sets results for selected models: 'DCS' - Document Context Sentences, 'TS-O' - Template Sentences with original entities, 'TS-R' - Template Sentences with replaced entities, 'RS' - Random Sentences. F1=multilabel F1-score, P=Precision, R=Recall, No=number of returned entities (lower is better). In the RS data set there are 2000 strings pretending to be sentences.

4 Discussion

On the basis of our research, we can draw a number of conclusions that are not often addressed to in publications about new neural models, their achievements and architecture. The scope of any assessment of new methods and models should be broadened to the understanding of their mistakes and the reasons why these models perform well or poorly in concrete examples, contexts and word meanings. These issues are particularly important in text data sets, in which semantic meaning and linguistic syntax are very complex.

In our effort to define linguistic categories for problematic Named Entities and their statistics in the CoNLL 2003 test set, we were able to draw a few additional conclusions regarding data annotation and augmentation processes. Moreover, our categories are similar to the taxonomy defined in publication about errors analysis for Uyghur Named Tagger (Abudukelimu et al., 2018).

4.1 The annotation process

The annotation process is a very tedious and exhaustive task for a person involved. Errors in data sets are expected but what must be checked is their impact on generalizing a model, e.g. one can create entities in places where they do not occur and check the model's stability. There are some useful applications for detecting annotation errors (Ratner et al., 2017), (Graliński et al., 2019) and (Wisniewski, 2018) but they are not used very often. Obviously, an appropriate and exhaustive documentation for the data set creation and annotation process is crucial. All annotated entity types should be described in details and examples of border cases should be given. In our analysis of the CoNLL 2003 data set we did not find any documentation. We have made our own assumptions and tried to guess why some classes are annotated in a given way. However, the work was hard and required many discussions and extended reviews of literature.

Secondly, there is a need for extended data sets with a broadened annotation process, similar to that of our diagnostic sets. E.g. linguists can extend their work not only just to the labelling of items (sentences), but also to indicating the scope of context that is necessary to recognise an entity, and to extending annotations for difficult cases or adding sub-types of entities.

Our work on diagnostic data sets is an attempt to extend an annotation process by focusing only on specific use cases which are less represented in the original data set.

4.2 Extended context

A new model training process itself should consist of more augmentation of the data set. Currently, there is some work being done on this topic, e.g. a semi-supervised context change with cutting the neighbourhood around NEs using a sliding window (Clark et al., 2018). Other techniques could be a random change of the first letter (or whole words) of NEs so that the model would not be so vulnerable to capitalized letters in names or small changes in sentences (e.g. adding or removing a dot at the end of a sentence).

Furthermore, a sentence itself is not always sufficient to recognise a class of a NE. In these cases, in both training and test data sets, there should be more samples where there are indications of co-references that are important to recognise particular NEs. Then, the input of a model should comprise a sentence and embedded features (or any representation) of co-references or their contexts. E.g. *"Little was banned. Peter Little took part in the last match with Welsh team."* - in the first sentence, we are are not sure if it is a NE. Then *"Peter Little"* indicates the proper NE type. An

example of a model and data processing pipeline (i.e. memory of embeddings) that takes into consideration the same names in different sentences is to be found in (Akbik et al., 2019) and (Zhang et al., 2018).

Another important improvement is adding information about document layout or the structure of a text, e.g. a table, its rows and columns, and headings. In CoNLL 2003, there are many sports news, stock exchange reports or timetables where the structure of a text helps to understand its context, and thus to better recognise its NEs. Such a solution for another domaininvoice information extractionis elaborated on by (Katti et al., 2018) or (Liu et al., 2019). The solutions mentioned here combine character information with document image information in one architecture of a neural network.

The CoNLL 2003 test set is certainly too small to test the generalisation and stability of a model. Faced with this issue, we must find new techniques to prevent over-fitting. For instance, we could check a model's resistance to examples prepared in our diagnostics data sets, e.g. after changing a NE in a template sentence, the model should find the entity in the same place. We could also prepare small modifications to our original sentences, e.g. add or remove a dot at the end of an example and compare results (similarly to adversarial methods).

5 Concluding remarks

Mistakes are not all created equal. A comparison of models based on scores like F1 is rather simplistic. In this paper we defined 4 major and 11 minor linguistic categories of errors for NER problems. For the CoNLL 2003 data set and five important ML models (Stanford, CMU, ELMO, FLAIR, BERT-base) we re-annotated all errors with respect to the newly proposed ontology.

The presented analysis helps better understand a source of problems in recent models and also to better understand why some models are more reliable on one data set but less not on another.

Acknowledgements

TSt, AWr, AW, DZi would like to thank for the financial support that their project conducted by Applica.ai has received from the European Regional Development Fund (POIR.01.01.01-00- 0144/17-00). PBi was financially supported by European Regional Development Fund POIR.01.01.01-00-0328/17.

Our research was also partially supported as a part of the RENOIR Project by the European Unions Horizon 2020 Research and Innovation Programme under the Marie Skodowska-Curie grant agreement No 691152 and by the Ministry of Science and Higher Education (Poland), grant Nos. W34/H2020/2016.

References

Halidanmu Abudukelimu, Abudoukelimu Abulizi, Boliang Zhang, Xiaoman Pan, Di Lu, Heng Ji, and Yang Liu. 2018. Error analysis of Uyghur name tagging: Language-specific techniques and remaining challenges. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics NAACL*. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649. Association for Computational Linguistics.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853.

Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. Cloze-driven pretraining of self-attention networks. *CoRR*, abs/1903.07785.

google bert. 2019. google-research/bert repository (issue 223).

Adrian Braşoveanu, Giuseppe Rizzo, Philipp Kuntschik, Albert Weichselbraun, and Lyndon J.B. Nixon. 2018. Framing named entity linking error types. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa.

2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Flair. 2018. Flair repository (issue 206 and 390).

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

Filip Graliński, Anna Wróblewska, Tomasz Stanisławek, Kamil Grabowski, and Tomasz Górecki. 2019. GEval: Tool for debugging NLP datasets and models. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 254–262, Florence, Italy. Association for Computational Linguistics.

Masaaki Ichihara, Kanako Komiya, Tomoya Iwakura, and Maiko Yamazaki. 2015. Error analysis of named entity recognition in bccwj.

Anoop R. Katti, Christian Reisswig, Cordula Guder, Sebastian Brarda, Steffen Bickel, Johannes Höhne, and Jean Baptiste Faddoul. 2018. Chargrid: Towards understanding 2d documents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4459–4469. Association for Computational Linguistics.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2018. A survey on deep learning for named entity recognition. *CoRR*, abs/1812.09449.

Xiaojing Liu, Feiyu Gao, Qiong Zhang, and Huasha Zhao. 2019. Graph convolution for multimodal information extraction from visually rich documents. *CoRR*, abs/1903.11279.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, CICLing'11, pages 171–189, Berlin, Heidelberg. Springer-Verlag.

MUC. 1995. Muc-6 challenges and data sets.

Christina Niklaus, Matthias Cetto, André Freitas, and Siegfried Handschuh. 2018. A survey on open information extraction. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3866–3878, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 78–86. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *CoRR*, abs/1711.10160.

Georges Rey. 2018. The analytic/synthetic distinction. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, fall 2018 edition. Metaphysics Research Lab, Stanford University.

Beth M. Sundheim. 1995. Overview of results of the muc-6 evaluation. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pages 13–31, Stroudsburg, PA, USA. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*.

Guillaume Wisniewski. 2018. Errator: a tool to help detect annotation errors in the universal dependencies project. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158. Association for Computational Linguistics.

Boliang Zhang, Spencer Whitehead, Lifu Huang, and Heng Ji. 2018. Global attention for name tagging. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 86–96.

Low-rank approximations of second-order document representations

Jarkko Lagus
Reaktor Innovations Oy,
University of Helsinki
jalagus@cs.helsinki.fi

Janne Sinkkonen
Reaktor Innovations Oy
janne.sinkkonen@reaktor.fi

Arto Klami
University of Helsinki
Department of Computer Science
arto.klami@cs.helsinki.fi

Abstract

Document embeddings, created with methods ranging from simple heuristics to statistical and deep models, are widely applicable. Bag-of-vectors models for documents include the mean and quadratic approaches (Torki, 2018). We present evidence that quadratic statistics alone, without the mean information, can offer superior accuracy, fast document comparison, and compact document representations. In matching news articles to their comment threads, low-rank representations of only 3–4 times the size of the mean vector give most accurate matching, and in standard sentence comparison tasks, results are state of the art despite faster computation. Similarity measures are discussed, and the Frobenius product implicit in the proposed method is contrasted to Wasserstein or Bures metric from the transportation theory. We also shortly demonstrate matching of unordered word lists to documents, to measure topicality or sentiment of documents.

1 Introduction

Today, most computational models for natural language are based on distributional representations. Words are routinely represented by *word embeddings* (Mikolov et al., 2013), most commonly as fixed-dimensional real-valued vectors, such as GloVe (Pennington et al., 2014) and fastText (Mikolov et al., 2018). Even though there is extensive literature on using, e.g., character-level and other sub-word information (Lee et al., 2017; Radford et al., 2017; Mikolov et al., 2018) or non-Euclidean embedding spaces (Nickel and Kiela, 2017; Muzellec and Cuturi, 2018), the standard embeddings remain currently as the default building block for practical tools.

Most applications do not, however, care about individual words. Instead, we may be concerned about the meaning of sentences or retrieval of documents, or in general, units larger than words. Distributed representations can be built for these larger units as well. Although sentences and documents differ as linguistic concepts, computational models for them can be similar when they are considered as sequences or even (unordered) bags of words.

Document representations often build on word embeddings. Already the mean of the word vectors turns out to be a surprisingly good representation (Wieting et al., 2015b), and accounting for the importance of words by a weighting scheme improves it further (Arora et al., 2016; Gupta et al., 2019). Even though the bare mean clearly ignores information, it is very efficient to compute. On the other end of the spectrum, document embeddings are built with computationally extremely heavy deep learning models such as ELMo (Peters et al., 2018), ULMFiT (Howard and Ruder, 2018), and BERT (Devlin et al., 2018). Deep models produce rich representations, but the amount of data and computation needed for training make them prohibitive for many applications.[1]

Our work falls between these two extremes. With the understanding that mean vectors may miss important aspects of documents, we want to develop fast and easy-to-use tools. This rules out complex deep networks. Instead, we focus on using second-order interactions between words, building on *covariance* of the embeddings of individual words, following the recent works of Torki (2018) and Nikolentzos et al. (2017).

The motivation of the paper is on finding fast and accurate ways to compare documents, or, alternatively, documents and semantics spanned by word lists. We start by evaluating document sim-

[1] For example BERT takes 0.5 secs to process a sentence on a CPU (Nvidia blog, our experiments), and getting good document representations may require fine-tuning.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 634–644
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

ilarity as pairwise similarities between words and show that this induces a compact approximative representation for the documents themselves. We relate the pairwise similarity to Wasserstein or Bures metric, used recently in various machine learning tasks (Arjovsky et al., 2017; Muzellec and Cuturi, 2018) and in quantum information theory (Bhatia et al., 2018).

The main result of these derivations is a practical document embedding strategy that builds on pre-trained word embeddings. The document embeddings are of relatively low dimension, larger than the word embeddings only by a small factor. They allow for efficient comparisons and are easy to implement and use in downstream tasks of document retrieval, sentence classification, etc. We demonstrate competitive performance against state-of-the-art methods in standard sentence similarity tasks (Conneau and Kiela, 2018), with a lower computational cost. We further demonstrate the approach in matching articles to their comment chains, and briefly in scoring moral sentiment and topicality defined by word lists.

2 Related work

Work on document representations has a long history in information retrieval. Sentence embeddings (Arora et al., 2016; Perone et al., 2018) is a related topic that has lately become more prominent, maybe because of the fast growth of social media platforms where communication is mostly done via short messages. For this paper, we treat sentences as short documents.

The mainline of research deals with building document vectors from pre-trained word vectors. The straightforward way averages over the word vectors. Wieting et al. (2015b) show that complex computations are not necessary for good document vectors. Instead, smart weighting under the averaging model is usually sufficient. On top of that work, weighting schemes and other heuristics have been proposed. The latest include common component removal (Arora et al., 2016), and the weighting schemes SIF (Arora et al., 2016), and P-SIF (Gupta et al., 2019), similar in idea to TF-IDF weighting. As alternatives to usage of pre-trained word embeddings, one can train directly document embeddings like skip-thought (ST) vectors (Kiros et al., 2015) basically generalizing the *word2vec* training method to sentences, or train the word embeddings and document embeddings together, but still within the bag-of-the-words averaging framework as is done with Paragraph Vectors (Le and Mikolov, 2014) and Doc2VecC (Chen, 2017).

Our core contributions are in the use of second-order information—covariance of the word vectors—for improving the representations. To our best knowledge, there is quite limited previous work in this direction. Torki (2018) used covariance matrices as (quite high-dimensional) representation for documents and Nikolentzos et al. (2017) represented documents with Gaussian distributions and used divergence metrics to compare the imposed distributions. We provide technical and computational analysis of the covariance approach, discuss similarity measures for the representations, including Frobenius and Wasserstein inner products, and show how low-rank approximations can then speed up the comparisons and make the representations more compact.

A completely different approach is taken by the deep learning community, with the use of universal language and transformer models such as ELMo (Peters et al., 2018), ULMFiT (Howard and Ruder, 2018), and BERT (Devlin et al., 2018). The accuracy of these deep learning models is state of the art, but the computational cost and need for training data are high. At the time of writing, there are no pre-trained models available for the Finnish language used in our experiments, and training such a base model would be costly as shown by Strubell et al. (2019).

3 Representations and similarities

Most applications compare word embeddings by the cosine similarity, $\cos(w_1, w_2) = \frac{w_1^T w_2}{|w_1||w_2|}$, where $w_1, w_2 \in \mathcal{R}^d$ are the embeddings (vectors). Cosine similarity is invariant to lengths of the vectors. Lengths typically do not encode semantics but relate to aspects like frequency of the word or homogeneity of its context.

Documents or sentences (later simply documents) are, in the absence of a sequence model, treated as bags of words. That is, methods for comparing documents are invariant to word order. We define for later purposes a *document matrix*,

$$D = \begin{bmatrix} \cdots & w_1 & \cdots \\ \cdots & w_2 & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & w_n & \cdots \end{bmatrix} \in \mathcal{R}^{n \times d}, \qquad (1)$$

as a collection of word vectors. Although the representation as such is not order-invariant, order-invariant ways to compare documents can be derived for this representation. The simplest is the cosine similarity of means, $\cos(1^T D_1/n_1, 1^T D_2/n_2)$. Performance of document mean vectors in benchmarks is not quite a state of the art, but decent enough for many applications (Wieting et al., 2015b).

A refinement from the simple average is to reweigh the word vectors before averaging, either in a general way or by specializing into the current corpus. Term-frequency inverse-document-frequency (TF-IDF) weighting is the classic way. Later variations of weighting schemes are smooth inverse frequency (SIF) (Arora et al., 2016) and its derivative partition SIF (P-SIF) (Gupta et al., 2019). They use weighted mean with weights computed from corpus; SIF uses $\alpha/(\alpha + p(w))$, where $\alpha > 0$ controls the smoothing based on empirical probability $p(w)$ of word w. The precomputation of weights for the whole corpus makes SIF unusable in some cases where the whole corpus is not available or is extremely large, and prohibits online processing. The issue is even more severe with P-SIF that requires more elaborate preprocessing.

For further improvement within the mean vector framework, it should be possible to improve performance either by (a) computing word embeddings in a way that optimizes the document embeddings (still computed as word vector averages), or by (b) transformations of the document averages in a way that takes the current document corpus into account. An example of the former is the Doc2VecC embedding (Chen, 2017), and SIF (Arora et al., 2016) demonstrates the latter by removing variation common to the corpus by projecting away the main variation over documents.

Still, a third way to improve performance would be to expand the representation from average while maintaining order invariance for model simplicity. This leads to second-order representations discussed next.

4 Second-order document representations

Our motivation arises from an empirical observation that mean vectors are not necessarily efficient summaries of documents (Fig. 1). At least with the *word2vec* embeddings, the distribution of word

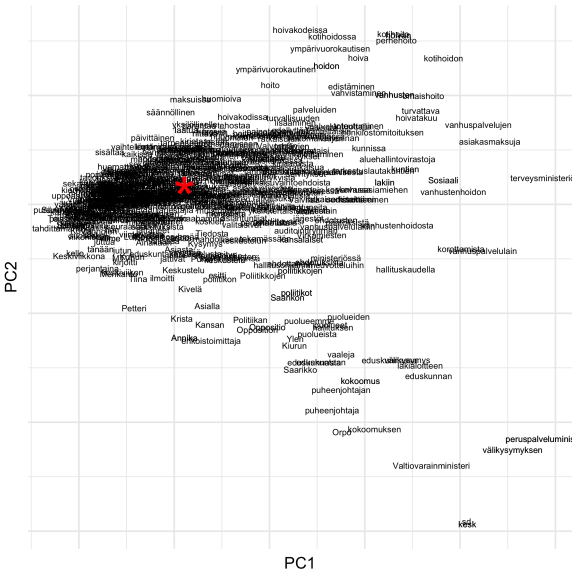


Figure 1: Words of a Finnish news article as *word2vec* vectors, projected to the 2D space of principal variation. The broad tail up right contains words related to health care and social services, especially those of seniors. The lower right tail is about politics. The mean vector (red star) makes a compromise between the tails, and is determined to a large degree by the numerous non-descriptive words projected closer to the origin. A representation that takes higher moments into account would catch the tails better. This structure seems to be typical to most of the documents in our corpora.

vectors (on the 2D-space of their principal variance) is strongly skewed, with a comet-like shape, often with more than one separate "tails". Mean is not an efficient statistic for such distribution, and it is an especially poor description of the tails that seem to carry the most descriptive words.

A representation based on higher-order moments would better catch the characteristics of the word vectors. Second moments, or covariance if centered to the means, would lose the asymmetry of the distributions, but the advantages compared to still higher-order moments are (relative) compactness and elegant algebra.

A question remains how second-order representations should be compared pairwise. A related worry is the computational efficiency, as the computational cost for naive second-order representations scales quadratically with the dimensionality of the embedding. Below, we first present an extension of cosine similarity to pairs of unordered collections of word vectors, and note how this induces a second-order representation for documents, and is interpretable as a Frobenius inner

product. We note a connection of this approach to existing work, compare to Bures and Wasserstein metric (later referred only as Wasserstein) and optimal transport theory, and later in the paper present empirical evidence of good performance of low-rank approximations of these representations.

4.1 Extending cosine similarity to documents

Using the notation of (1), let $A \in \mathcal{R}^{n\times d}, B \in \mathcal{R}^{m\times d}$ be the word vectors of two documents, sentences, or other unordered collections of words, with word counts (n, m) and embedding of dimensionality d.

The core of cosine similarity of single words a, b is the ordinary inner product $\cos(a, b) \propto a^T b$, with a normalization such that $\cos(a, a) = 1$ for any $a \neq 0$. We generalize this to similarities of word collections A and B, as the normalized square sum of all pairwise dot products:

$$\begin{aligned} Z(A,B)\, h^2(A,B) \equiv \\ \sum_{a,b}(a^T b)^2 = \sum_{a,b} ab^T\, ba^T \\ = \sum_a aB^T Ba^T = \sum_a \mathrm{Tr}(Ba^T aB^T) \\ = \mathrm{Tr}(BA^T AB^T) = \mathrm{Tr}(A^T AB^T B) \end{aligned}$$

with $Z(A, B)$ such that $h^2(A, A) = 1$ for any $A \neq 0$. The unnormalized trace structure appears so often later in the paper, that we define a shorter notation for it:

$$A * B \equiv [\mathrm{Tr}(BA^T AB^T)]^{1/2}. \quad (2)$$

The trace is interpretable as the *Frobenius inner product* of covariance-like but unnormalized matrices $C_A \equiv A^T A$ and $C_B \equiv B^T B$:

$$(A * B)^2 = \mathrm{Tr}(C_A^T C_B)\,. \quad (3)$$

With normalization, the similarity then becomes

$$h^2(A, B) = \frac{(A * B)^2}{(A * A)\,(B * B)} \equiv A' * B'\,, \quad (4)$$

where the last expression prenormalizes the word list matrices so that

$$A' \equiv A/(A * A)^{1/2}\,. \quad (5)$$

Finally, one can centralize word matrices before computing the similarity, $A_0 = A - 1^T m_A$, etc., without affecting the formalism:

$$h(A_0, B_0) = A_0' * B_0'\,. \quad (6)$$

Torki (2018) introduced a document representation named DoCoV and extended it in experiments to include mean vector gaining an increase in performance. This variant of the DoCoV representation is defined as $\mathrm{vec}(m_A, A_0^T A_0)$, where the vec operation concatenates the arguments and flattens the matrices row by row. This is similar to our cross product C_A but includes the mean. For computing similarities they used dot products

$$\begin{aligned} m_A^T m_B + vec(A_0^T A_0)^T\, vec(B_0^T B_0) \\ = m_A^T m_B + (A_0 * B_0)^2\,. \end{aligned}$$

We note that this is the similarity h^2 introduced above, summed to the cosine similarity of the means, but the normalization of the dot product by Torki (2018) is for the entire concatenated vectors, which is reasonable as long as mean and covariance are treated as commensurable. There is a unit inconsistency in the concatenation, for the covariance term is quadratic while the mean is not. Our reinterpretation and slight reformulation of the similarity as h^2 offers a way of substantially shrinking the document representations (Section 5), and including the mean does not seem empirically important.

4.2 Connection to transport theory

Wasserstein metric compares two (elliptic or gaussian) distributions defined by their means and covariances. It emerges in the optimal transport theory as a measure of minimum-path transport of the probability mass of distribution (m_A, C_A) to distribution (m_B, C_B):

$$\begin{aligned} \mathcal{W}^2((m_A, C_A), (m_B, C_B)) = \\ ||m_A - m_B||^2 + \\ \mathrm{Tr}\left(C_A + C_B - 2\left(C_A^{1/2}\, C_B\, C_A^{1/2}\right)^{1/2}\right)\,. \end{aligned}$$

For covariances computed from word vectors, $C_A = A^T A$, etc., we would like to have scale invariance similar to cosine or the Frobenius cosine above. But it is not clear how to obtain scale invariance in a principled way, for the scales of terms $\mathrm{Tr}\, C_A$ and $\mathrm{Tr}\, C_B$ vary differently from the scale of the cross term.

Within the context of their elliptical embeddings, Muzellec and Cuturi (2018) define a Bures or Wasserstein cosine by normalizing the sole

cross term[2]:

$$h_{\mathcal{W}}(C_A, C_B) = \frac{\operatorname{Tr}\left(C_A^{1/2} C_B C_A^{1/2}\right)^{1/2}}{[\operatorname{Tr} C_A]^{1/2} [\operatorname{Tr} C_B]^{1/2}} .$$

This form is surprisingly close to the "Frobenius cosine" of Eq. 4, and allows prenormalization of representations: First by applying the cyclic property of trace, and then moving normalizations to be computed first, we have

$$\begin{aligned} h_{\mathcal{W}}(C_A, C_B) &= \frac{\operatorname{Tr}\left(C_A^T C_B\right)^{1/2}}{[\operatorname{Tr} C_A]^{1/2} [\operatorname{Tr} C_B]^{1/2}} \\ &= \operatorname{Tr}\left(\left[\frac{C_A}{\operatorname{Tr} C_A}\right]^T \left[\frac{C_B}{\operatorname{Tr} C_B}\right]\right)^{1/2} . \end{aligned}$$

The difference to our similarity defined in Eq. 4 is only in the order of the outer square root and trace operators. If one denotes the eigenvalues of the suitably normalized matrix $C_A C_B$ by η_i^2, the Wasserstein cosine is equal to $\sum_i \eta_i$, while the Frobenius cosine equals to $(\sum_i \eta_i^2)^{1/2}$. The latter obviously gives more weight to "high-variance covariation" of C_A and C_B, that is, for larger eigenvalues. When only one eigenvalue is non-zero, the cosines are equal.

Matrix square root in the Wasserstein cosine, however, seems to require matrix diagonalization for every document comparison, which would be of computational complexity $O(d^3)$.

So although deriving anything equivalent to Frobenius cosine by starting from the Wasserstein metrics seems hard, similarities may be worth further investigation, either empirical or theoretical. An empirical comparison, presented in Figure 2, suggests that in practice the two cosines are quite closely related. Experiments later in the paper indicate better practical performance from the Frobenius cosine, in some applications. Frobenius cosine is notably faster to compute in practice, as shown in the next section.

[2]Or actually they have two separately normalized terms, one for means as ordinary cosine, and one for quadratics. We test the two-term version of our Frobenius product briefly in the experimental section and find that the mean term does not help performance there. The supplementary material of Muzellec and Cuturi (2018) easily leads to a similar conclusion with the Wasserstein cosine.

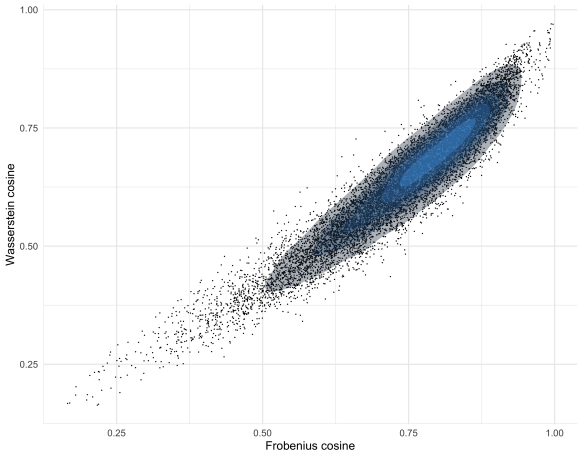


Figure 2: For our document collection of Finnish news articles and their comments, the Frobenius and Wasserstein cosines are closely related with correlation coefficient $r = 0.95$.

5 Computation and low-rank approximation

The Frobenius cosine of Eq. 4 can be computed by using word matrices A (of size $n \times d$) as document representations. Comparing these by pairwise vector inner products (Eq. 2) would have computationally cost of $O(n_A n_B d)$ — efficient for short word lists but not when lengths n approach or exceed d.

On the other hand, one can follow in the footsteps of Torki (2018) and represent documents as covariance-like inner products $A^T A$, which are of dimensionality $d(d+1)/2$. Torki (2018) named this method as DoCoV descriptor. In this case, the Frobenius complexity would then be $O(d^2)$, which for long documents is cheaper than a pairwise comparison of word lists, but still expensive compared to inner products of mean vectors, $O(d)$.

Key to low-order approximations is to note that the rows of the word lists A etc. do not need to represent words: Instead, any vectors $\hat{A}$ would give the same Frobenius product (and cosine) as long as the covariances are preserved, that is, $\hat{A}^T \hat{A} = A^T A$. If $\hat{A}$ is just an approximation of A, of, say, dimensionality $k \times d$ and of similar size for all documents, the pairwise Frobenius computation (Eq. 2) would be of complexity $O(k^2 d)$.

Our approximation and computation strategy is therefore to replace A with a suitable approximation $\hat{A}$, and compute the Frobenius inner product

without ever realizing the covariance matrices, by

$$\hat{A} * \hat{B} = \left(\sum_{kl} (\hat{A})^T_{k\cdot} (\hat{B})_{l\cdot} \right)^{1/2} . \quad (7)$$

For prenormalized approximations (Eq. 5), this is directly the desired similarity h^2 (as in Eq. 4).

We cannot optimize for approximation errors of pairwise Frobenius cosines, so we choose to optimize representations so that $A * A = \mathrm{Tr}(A^T A)$ is well preserved. Remembering that, for a symmetric matrix, trace is the sum of its eigenvalues, we may specifically choose a decomposition $H^T H$ of $A^T A$ such that dropping rows from H minimizes the approximation error in $\mathrm{Tr}(A^T A)$. The optimal H consists of the eigenvectors of $A^T A$ multiplied by square roots of their eigenvalues:

$$A^T A = U^T \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} U = (U\Lambda^{\frac{1}{2}})^T (U\Lambda^{\frac{1}{2}}) \equiv H^T H .$$

Now $\mathrm{Tr}(A^T A) = \sum_i \lambda_i$, where λ are the diagonal of Λ, the eigenvalues. All eigenvalues of a positive semidefinite matrix are non-negative, so trace is best approximated with a $\hat{H}$ that contains the eigenvectors associated to largest eigenvalues (the square roots of the eigenvalues multiplied in):

$$\hat{A} = \hat{H} = \begin{bmatrix} \sqrt{\lambda_1} u_1 \\ \sqrt{\lambda_2} u_2 \\ \cdots \\ \sqrt{\lambda_3} u_k \end{bmatrix} \in \mathcal{R}^{k \times d} . \quad (8)$$

Let the SVD of the word list matrix be $A = U\Lambda' V^T$. Then $A^T A = V\Lambda' U^T U \Lambda' V^T = V\Lambda'^2 V^T$, the last form being the eigenvalue decomposition of $A^T A$. So the approximation $\hat{H}$ can be obtained directly from the SVD of the word list A, without computing the covariance matrix. This is relatively cheap for small ranks k, and can be scaled up for documents with very large n with stochastic methods (Halko et al., 2011). The complexity depends on the SVD algorithm chosen.

Note that analogously to the original word lists, the above approximation is applicable to centered word lists A_0, and the normalized counterparts A' and A'_0, as long as normalization of representations is done after approximation (to preserve $h^2(\hat{A}', \hat{A}') = 1$).

The Frobenius cosine can, therefore, be efficiently computed with Eq. 7 of complexity $O(k^2 d)$, if the documents are approximated by the first k principal vectors of the word list SVD. Savings, compared to full covariances, are of order k/d for space, and k^2/d for document comparison times.

6 Method summary

The pre-computation process for online comparison document comparison or search is as follows:

1. Choose suitable word embeddings for the corpus, defining factors being language and the domain.

2. Collect word vectors of each document into matrix A, as in Eq. 1.

3. Choose a rank k, typically 2–20. Compute the k-rank (SVD left side) approximation $\hat{A}_0$ of centered documents like in Eq. 8. Prenormalize (Eq. 5) and store $\hat{A}'_0$.

4. Compare documents by $\hat{A}'_0 * \hat{B}'_0$, using the pairwise dot products as in Eq. 7 for computation.[3]

Representations for new, upcoming documents repeat steps 2–3. (There is no global model to update or refer to.)

7 Experiments

To validate the proposed representations and the similarity h^2, we conducted two separate experiments and one demonstration. In the first one, proposed methods are compared to state-of-the-art sentence embedding models on the tasks of Facebook's SentEval library (Conneau and Kiela, 2018). The other experiment and the demonstration apply the techniques to news articles and their comments on three Finnish media sites. For the latter experiment, 88,986 articles with comments were gathered from the associated websites.

For English we use fastText (Mikolov et al., 2018) embeddings with $d = 300$, but note that similar results were achieved also with GloVe (Pennington et al., 2014) and Word2Vec (Mikolov et al., 2013). For Finnish, we use *word2vec* embeddings, provided by the Turku NLP group[4], since fastText does not provide adequate embeddings for Finnish.

[3] With rank $k = 1$, the similarity h^2 is equal to square of the cosine between directions of principal variation. The principal vectors have no well-defined polarity, so taking a square or absolute value of the cosine is important.

[4] Older version of the *fin-word2vec.bin* embeddings with dimension 300, linked from http://bionlp.utu.fi/finnish-internet-parsebank.html.

7.1 Textual similarity tasks

As an initial experiment on the quality of the second-order representations, we evaluated them with the standard unsupervised similarity tasks (STS12 - STS16) using SentEval library by Facebook (Conneau and Kiela, 2018).

We compare a few different low-rank approximations using our proposed method against the state-of-the-art unsupervised and semi-supervised methods in Table 1. We see that already at $k = 10$, the approximation reaches close to the accuracy of the full rank, within this dataset, while decreasing the size of representations significantly (here 300×10 vs. 300×300). It surpasses most of the other unsupervised methods and compares well with the semi-supervised methods. Interestingly, already the rank-1 approximation is better than the classical mean vector approach (Glove Avg in Table 1), while having the same computational complexity.

Our low-rank estimation is on par with the related unsupervised method, DoCoV, and the only model reaching clearly higher scores is the P-SIF + PSL by Gupta et al. (2019), which uses computationally relatively heavy reweighting.

Figure 3 demonstrates the validity of the low-rank approximation, by plotting the STS evaluation scores against the rank k of the approximation. The results are as expected: Using more components retains more information, and $k \approx 5 \ldots 10$ is roughly enough for all tasks. Contrary to the experiment of the next section, using too large k does not hurt performance (except nominally in some cases).[5]

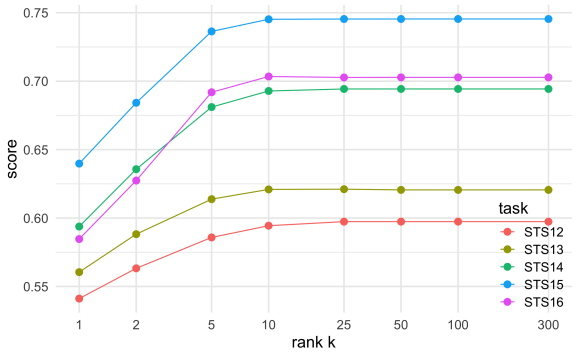


Figure 3: STS scores vs. rank k of the covariance approximation. Performance on these sentence tasks mostly saturates at $k = 5 \ldots 25$.

7.2 Comments vs. article

Many news sites contain a comment section associated with articles. It can be useful to compare articles to the comments, for example for moderation: If the discussion drifts too far from the original topic, human attention may be needed. In this experiment, we tried to find real article–comment pairs from a set in which half of the pairs were fake, as a proxy for the moderation task. It tests the semantic resolution of the similarity measures and has the convenience of known ground truth.

We crawled articles and their comment sections from three Finnish news sources: Yle (national broadcasting company), Uusi Suomi (news and blogging platform), and Iltalehti (a tabloid). Comments were concatenated to a single document (per article), retaining all article–comment pairs with at least 100 words on both sides. This gave 16,263, 18,548, and 9,682 pairs for Iltalehti, Uusi Suomi and Yle, respectively. The pairs were concatenated to sets of fake pairs of equal size, obtained by permuting the real pairs. We then rank the pairs according to decreasing similarity, and consider real pairs positive instances for retrieval measures. Figure 4 shows the ROC-like curves for our proposed h^2 of Eq. 7 (with rank $k = 4$), and two baselines based on document means, one computed directly from the word vectors and one after SIF weighting. The proposed method performs clearly better than average-based methods for all news sources, demonstrating the benefits of including second moments. It also outperforms the Wasserstein cosine $h_{\mathcal{W}}$ while having a clear advantage also in computation speed, and centering improves the results.

To see the effect of the rank, we ran the same test for various values of k and evaluated the recall at the selected value of precision (5% of fake pairs retrieved). All ranks $k > 0$ outperform the baseline of mean-vector cosine (horizontal line), except for Uusi Suomi at $k = 1$ (Figure 5). A bit unexpectedly, optimum is already at low values of k, which is nice from the computational perspective and suggests a regularization effect from SVD maybe worth of further study.

Finally, we wanted to see if adding a conventional mean term to the similarity computed with centralized second-order terms only helps with resolution, and ran the tests with various values of the weight α for the second-order term. One

[5] This may be because of the relative shortness of sentences here, vs. documents in the other experiment. The short sentences cannot even span a high-dimensional representation.

	Unsupervised					Semi-supervised			Ours		
Task	ST	Glove Avg	Glove tf-idf	DoCoV	P-SIF PSL	PSL Avg	Glove WR	PSL WR	Frob. $k = 1$	Frob. $k = 10$	Frob. $k = 300$
STS12	30.8	52.5	58.7	56.4	65.7	52.8	56.2	59.5	54.1	59.4	59.7
STS13	24.8	42.3	52.1	62.1	64.0	46.4	56.6	61.8	56.0	62.1	62.1
STS14	31.4	54.2	63.8	70.3	74.8	59.5	68.5	73.5	59.4	69.3	69.4
STS15	31.0	52.7	60.6	76.2	77.3	60.0	71.7	76.3	64.0	74.5	74.6
STS16	51.4	47.2	51.1	73.0	73.7	63.3	72.4	72.5	58.5	70.3	70.3

Table 1: Our method vs. other unsupervised and semi-supervised methods (from Gupta et al., 2019) on semantic textual similarity (STS) tasks, evaluated as Pearson correlations to human ground truth. ST stands for skip-thought vectors (Kiros et al., 2015), WR for SIF weighting combined with common component removal (Arora et al., 2016), and PSL for PARAGRAM-SL999 word vectors (Wieting et al., 2015a).

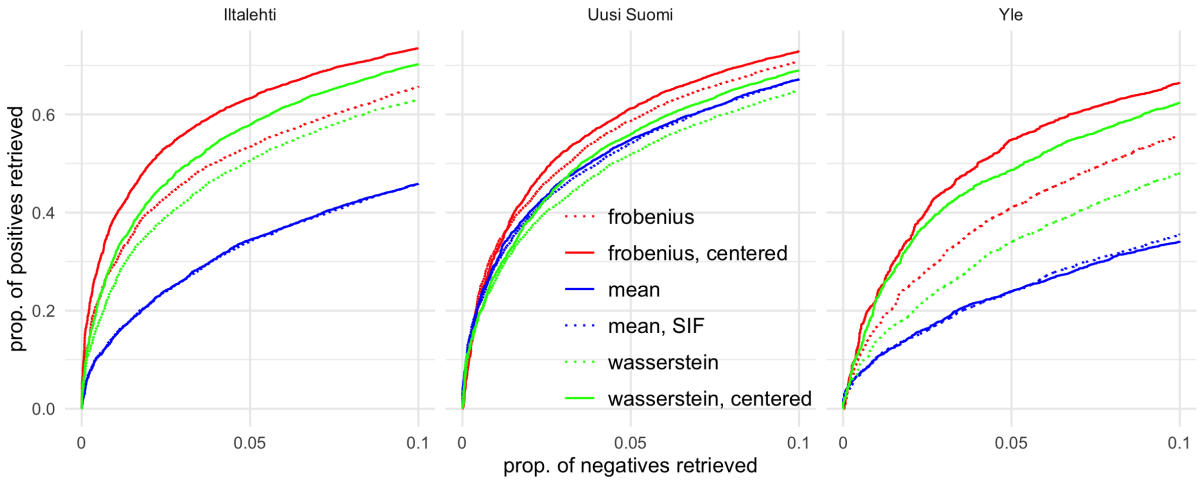


Figure 4: Fake vs. real pairs retrieved (ROC curves), in the article–comment matching tasks, for the three news sources, and for different matching methods. Cosine of mean vectors with original *word2vec* weights (blue; solid) and SIF weights (blue; dotted) perform practically identically. Quadratic approaches are all better, in general Frobenius (red) outperforming Wasserstein (green), and centered covariances (solid) outperforming non-centered ones (dotted). More fine-grained evaluations over k and mean-vector cosine mixing (Fig. 5 and 6) are run for 5% of the fake pairs retrieved (at the middle of x-axis).

source (Uusi Suomi) peaks around $\alpha = 0.5$, but for other sources, the performance increases monotonically with the weight of h^2. Apparently, mean vector is at least sometimes redundant if just low-rank second-order information is available.[6]

7.3 Media segmentation

Finally, the classic task of scoring documents for sentiment or topic by a word list is amenable to the application of the Frobenius similarity h^2. The target word list is usually just an unordered collection of words, although it may be weighted. Extensive sets of word lists are curated, for example, by LIWC. Likely, the semantics of such a list would sometimes be better operationalized by the suggested quadratic representation rather than by the list itself or its vectorized mean (with respect to an embedding).

As an example, we just present a single finding in Figure 7. The moral content of the comment chains of online news articles seems to vary by source, and climate change as a topic has differing effects, depending on the platform. The moral word lists were manually augmented and translated from the lists available from the developers of the Moral Foundations Theory (MFT)[7].

8 Conclusions

As already demonstrated by Torki (2018) and Nikolentzos et al. (2017), taking second moments of word vectors into document represen-

[6] The appendix of (Muzellec and Cuturi, 2018) gives similar impression for Wasserstein cosine: The performance is relatively flat over wide range of α and sometimes seems to increase monotonically.

[7] https://www.moralfoundations.org/othermaterials

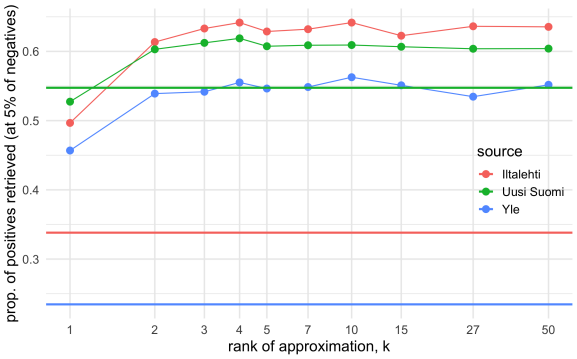


Figure 5: Frobenius recall of the article–comment matching task (proportion of real pairs retrieved when 5% of fake pairs are retrieved), as a function of rank k. For all news sources, performance saturates at rather low orders, $k = 2 \ldots 10$. Horizontal lines indicate recall with cosine of mean word vectors, which ignores second moments of the document.

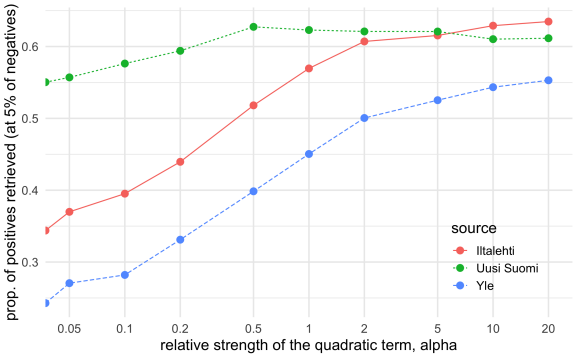


Figure 6: A recall measure on the article–comment matching task (proportion of real pairs retrieved when 5% of fake pairs are retrieved), when matching is with a mixture $\cos + \alpha h^2$ of ordinary cosine of means and centered Frobenius similarity (with rank $k = 3$). Adding mean information (smaller α) generally degrades performance except for one news source (*Uusi Suomi*) which peaks at $\alpha \approx 0.5$.

tations improves document matching. Surprisingly, the often-used mean vector seems to then become about irrelevant, a finding that needs to be replicated with other embeddings and larger experiments. There may exist efficient, ICA-like schemes relying on even higher moments to optimize the representations.

The second-order representations, and an associated similarity measure equivalent to the Frobenius inner product, can be derived by extending the Euclidean inner product into sets of words in a natural, pairwise manner. Empirically, the Frobenius similarity closely approximates Wasserstein similarity, familiar from transport theory, but allows efficient low-rank approximations of otherwise high-dimensional representations. The relationships of the two similarities may be worth further investigation, both theoretical and empirical.

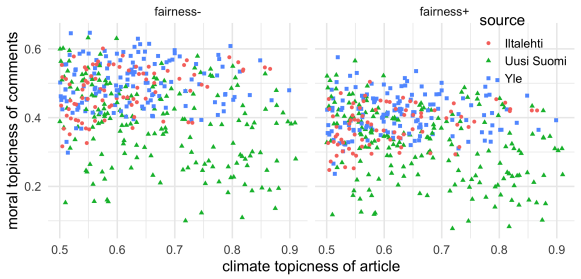


Figure 7: Moral sentiment of text, in the sense of the Moral Foundation Theory (Graham et al., 2013), has been measured by counting words appearing on a curated list (Garten et al., 2016). Moral word lists can also be related to documents with our h^2. On a blogging platform (*Uusi Suomi*), climate change as a prominent topic does not always rise moral comments at all (comments are probably technical). Points represent single article–comment pairs for articles of high topicality. Left: fairness, negative polarity; right: fairness, positive polarity.

Low-rank approximations are not only computationally useful, but also may sometimes have a regularizing effect that improves matching. Like mean vectors, second-order representations are useful as an alternative to traditional word-occurrence scoring, on quantifying sentiment and topicality of documents. Our experiments also show that rank-1 approximations are better representations of the documents than the mean vectors, while having the same representation size and computational complexity. There is an interesting contrast to the SIF preprocessing, where one step is to remove the corpus-wide largest component of variance to enhance the performance. While these two results are not contradictory, the combination is somewhat counter-intuitive.

Compared to other embedding methods, our approach requires only low precomputation effort and is local to document. The locality allows online processing that would be hard to implement with methods requiring preprocessing the corpus. If the online property is not needed, second-order representations are compatible with smart weighting schemes like SIF (Arora et al., 2016) or P-SIF (Gupta et al., 2019), and also with corpus-wide preprocessing schemes like projections and scalings.

References

Martin Arjovsky, Soumith Chintala, and Lon Bottou. 2017. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Rajendra Bhatia, Tanvi Jain, and Yongdo Lim. 2018. On the Bures–Wasserstein distance between positive definite matrices. *Expositiones Mathematicae*.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. *arXiv preprint arXiv:1707.02377*.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Justin Garten, Reihane Boghrati, Joe Hoover, Kate M Johnson, and Morteza Dehghani. 2016. Morality between the lines: Detecting moral sentiment in text. In *Proceedings of IJCAI 2016 workshop on Computational Modeling of Attitudes*.

Jesse Graham, Jonathan Haidt, Sena Koleva, Matt Motyl, Ravi Iyer, Sean P Wojcik, and Peter H Ditto. 2013. Moral foundations theory: The pragmatic validity of moral pluralism. In *Advances in experimental social psychology*, volume 47, pages 55–130. Elsevier.

Vivek Gupta, Ankit Kumar Saw, Partha Pratim Talukdar, and Praneeth Netrapalli. 2019. Unsupervised Document Representation using Partition Word-Vectors Averaging.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Boris Muzellec and Marco Cuturi. 2018. Generalizing point embeddings using the wasserstein space of elliptical distributions. In *Advances in Neural Information Processing Systems*, pages 10258–10269.

Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pages 6338–6347.

Giannis Nikolentzos, Polykarpos Meladianos, François Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2017. Multivariate gaussian document representation from word embeddings for text categorization. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 450–455.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Christian S. Perone, Roberto Silveira, and Thomas S. Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to Generate Reviews and Discovering Sentiment. *CoRR*, abs/1704.01444.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. *arXiv preprint arXiv:1906.02243*.

Marwan Torki. 2018. A Document Descriptor using Covariance of Word Vectors. In *Proceedings of the*

56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 527–532.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015a. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015b. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Named Entity Recognition with Partially Annotated Training Data

Stephen Mayhew[♮], **Snigdha Chaturvedi**[♯], **Chen-Tse Tsai**[♭], **Dan Roth**[♮]
[♮]University of Pennsylvania, Philadelphia, PA, 19104
[♯]University of North Carolina, Chapel Hill, NC, 27599
[♭]Bloomberg LP
{mayhew, danroth}@seas.upenn.edu,
snigdha@cs.unc.edu, ctsai54@bloomberg.net

Abstract

Supervised machine learning assumes the availability of fully-labeled data, but in many cases, such as low-resource languages, the only data available is partially annotated. We study the problem of Named Entity Recognition (NER) with partially annotated training data in which a fraction of the named entities are labeled, and all other tokens, entities or otherwise, are labeled as non-entity by default. In order to train on this noisy dataset, we need to distinguish between the true and false negatives. To this end, we introduce a constraint-driven iterative algorithm that learns to detect false negatives in the noisy set and downweigh them, resulting in a weighted training set. With this set, we train a weighted NER model. We evaluate our algorithm with weighted variants of neural and non-neural NER models on data in 8 languages from several language and script families, showing strong ability to learn from partial data. Finally, to show real-world efficacy, we evaluate on a Bengali NER corpus annotated by non-speakers, outperforming the prior state-of-the-art by over 5 points F1.

1 Introduction

Most modern approaches to NLP tasks rely on supervised learning algorithms to learn and generalize from labeled training data. While this has proven successful in high-resource scenarios, this is not realistic in many cases, such as low-resource languages, as the required amount of training data just doesn't exist. However, partial annotations are often easy to gather.

We study the problem of using partial annotations to train a Named Entity Recognition (NER) system. In this setting, all (or most) identified entities are correct, but not all entities have been identified, and crucially, there are no reliable examples of the negative class. The sentence shown in Figure 1 shows examples of both a gold and a partially annotated sentence. Such partially annotated data is relatively easy to obtain: for example, a human annotator who does not speak the target language may recognize common entities, but not uncommon ones. With no reliable examples of the negative class, the problem becomes one of estimating which unlabeled instances are true negatives and which are false negatives.

			False Negative					
	Arsenal	hires	Unai	Emery	as	Arsene	Wenger's	successor
Gold	ORG	O	PER	PER	O	PER	PER	O
Partial	ORG	O	O	O	O	PER	PER	O
Weights								
Oracle	1	1	0	0	1	1	1	1
Our model	1	0.6	0.01	0.03	0.8	1	1	0.4

Figure 1: This example has three entities: *Arsenal*, *Unai Emery*, and *Arsene Wenger*. In the *Partial* row, the situation addressed in this paper, only the first and last are tagged, and all other tokens are assumed to be non-entities, making *Unai Emery* a false negative as compared to *Gold*. Our model is an iteratively learned binary classifier used to assign weights to each token indicating its chances of being correctly labeled. The *Oracle* row shows optimal weights.

To address the above-mentioned challenge, we present Constrained Binary Learning (CBL) – a novel self-training based algorithm that focuses on iteratively identifying true negatives for the NER task while improving its learning. Towards this end, CBL uses constraints that incorporate background knowledge required for the entity recognition task.

We evaluate the proposed methods in 8 languages, showing a significant ability to learn from partial data. We additionally experiment with initializing CBL with domain-specific instance-weighting schemes, showing mixed results. In the process, we use weighted variants of popular NER models, showing strong performance in both non-neural and neural settings. Finally, we show experiments in a real-world setting, by employing non-speakers to manually annotate romanized Bengali text. We show that a small amount of non-speaker annotation combined with our method can outperform previous methods.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 645–655
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

2 Related Work

The supervision paradigm in this paper, partial supervision, falls broadly under the category of semi-supervision (Chapelle et al., 2009), and is closely related to weak supervision (Hernández-González et al., 2016)[1] and incidental supervision (Roth, 2017), in the sense that data is constructed through some noisy process. However, all of the most related work shares a key difference from ours: reliance on a small amount of fully annotated data in addition to the noisy data.

Fernandes and Brefeld (2011) introduces a transductive version of structured perceptron for partially annotated sequences. However, their definition of partial annotation is labels removed at random, so examples from all classes are still available if not contiguous.

Fidelity Weighted Learning (Dehghani et al., 2017) uses a teacher/student model, in which the teacher has access to (a small amount) of high quality data, and uses this to guide the student, which has access to (a large amount) of weak data.

Hedderich and Klakow (2018), following Goldberger and Ben-Reuven (2017), add a noise adaptation layer on top of an LSTM, which learns how to correct noisy labels, given a small amount of training data. We compare against this model in our experiments.

In the world of weak supervision, Snorkel (Ratner et al., 2017; Fries et al., 2017), is a system that combines automatic labeling functions with data integration and noise reduction methods to rapidly build large datasets. They rely on high recall and consequent redundancy of the labeling functions. We argue that in certain realistic cases, high-recall candidate identification is unavailable.

We draw inspiration from the Positive-Unlabeled (PU) learning framework (Liu et al., 2002, 2003; Lee and Liu, 2003; Elkan and Noto, 2008). Originally introduced for document classification, PU learning addresses problems where examples of a single class (for example, sports) are easy to obtain, but a full labeling of all other classes is prohibitively expensive.

Named entity classification as an instance of PU learning was introduced in Grave (2014), which uses constrained optimization with constraints similar to ours. However, they only address the problem of named entity classification, in which mentions are given, and the goal is to assign a *type* to a named-entity (like 'location', 'person', etc.) as opposed to our goal of identifying and typing named entities.

Although the task is slightly different, there has been work on building 'silver standard' data from Wikipedia (Nothman et al., 2008, 2013; Pan et al., 2017), using hyperlink annotations as the seed set and propagating throughout the document.

Partial annotation in various forms has also been studied in the contexts of POS-tagging (Mori et al., 2015), word sense disambiguation (Hovy and Hovy, 2012), temporal relation extraction (Ning et al., 2018), dependency parsing (Flannery et al., 2012), and named entity recognition (Jie et al., 2019).

In particular, Jie et al. (2019) study a similar problem with a few key differences: since they remove entity surfaces randomly, the dataset is too easy; and they do not use constraints on their output. We compare against their results in our experiments.

Our proposed method is most closely aligned with the Constraint Driven Learning (CoDL) framework (Chang et al., 2007), in which an iterative algorithm reminiscent of self-training is guided by constraints that are applied at each iteration.

3 Constrained Binary Learning

Our method assigns instance weights to all negative elements (tokens tagged as O), so that false negatives have low weights, and all other instances have high weights. We calculate weights according to the confidence predictions of a classifier trained iteratively over the partially annotated data. We refer to our method as Constrained Binary Learning (CBL).[2]

We will first describe the motivation for this approach before moving on to the mechanics. We start with partially annotated data (which we call set T) in which some, but not all, positives are annotated (set P), and no negative is labeled. By default, we assume that any instance not labeled as positive is labeled as negative as opposed to unlabeled. This data (set N) is noisy in the sense that many true positives are labeled as negative (these are *false negatives*). Clearly, training on T as-is will result in a noisy classifier.

Two possible approaches are: **1)** find the false negatives and label them correctly, or **2)** find the false negatives and remove them. The former method affords more training data, but runs the risk of adding noise, which could be worse than the original partial annotations. The latter is more forgiving because of an asymmetry in the penalties: it is important to remove all false negatives in N, but inadvertently removing true negatives from N is typically not a problem, especially in NER, where negative examples dominate. Further, a binary model (only two labels) is sufficient in this case, as we need only detect entities, not type them.

We choose the latter method, but instead of removing false negatives, we adopt an instance-weighting approach, in which each instance is assigned a weight $v_i \geq 0$ according to confidence in the labeling of that instance. A weight of 0 means that the loss this instance incurs during training will not update the model.

With this in mind, CBL takes two phases: first, it learns a binary classifier λ using a constrained iterative process modeled after the CODL framework (Chang et al., 2007), and depicted in Figure 2. The core of

[1] See also: https://hazyresearch.github.io/snorkel/blog/ws_blog_post.html

[2] Publication details (including code) can be found at cogcomp.org/page/publication_view/888

Require:
P : positive tokens
N : noisy negative tokens
C : constraints

1: $T = N \cup P$
2: $V \leftarrow$ Initialize T with weights (Optional)
3: **while** stopping condition not met **do**
4: $\quad \lambda \leftarrow \text{train}(T, V)$
5: $\quad \hat{T} \leftarrow \text{predict}(\lambda, T)$
6: $\quad T, V \leftarrow \text{inference}(\hat{T}, C)$
7: **end while**
8: return λ

Figure 2: Constrained Binary Learning (CBL) algorithm (phase 1). The core of the algorithm is in the while loop, which iterates over training on T, predicting on T and correcting those predictions.

the algorithm is the train-predict-infer loop. The training process (line 4) is weighted, using weights V. At the start, these can be all 1 (Raw), or can be initialized with prior knowledge. The learned model is then used to predict on all of T (line 5). In the inference step (line 6), we take the predictions from the prior round and the constraints C and produce a new labeling on T, and a new set of weights V. The details of this inference step are presented later in this section. Although our ultimate strategy is simply to assign weights (not change labels), in this inner loop, we update the labels on N according to classifier predictions.

In the second phase of CBL, we use the λ trained in the previous phase to assign weights to instances as follows:

$$v_i = \begin{cases} 1.0 & \text{if } x_i \in P \\ P_\lambda(y_i = \text{O} \mid x_i) & \text{if } x_i \in N \end{cases} \quad (1)$$

Where $P_\lambda(y_i = \text{O} \mid x_i)$ is understood as the classifier's confidence that instance x_i takes the negative label. In practice it is sufficient to use any confidence score from the classifier, not necessarily a probability. If the classifier has accurately learned to detect entities, then for all the false negatives in N, $P_\lambda(y_i = \text{O}|x_i)$ is small, which is the goal.

Ultimately, we send the original multiclass partially annotated dataset along with final weights V to a standard weighted NER classifier to learn a model. No weights are needed at test time.

3.1 NER with CBL

So far, we have given a high-level view of the algorithm. In this section, we will give more low-level details, especially as they relate to the specific problem of NER. One contribution of this work is the inference step (line 6), which we address using a constrained Integer Linear Program (ILP) and describe in this section. However, the constraints are based on a value we call the *entity ratio*. First, we describe the entity ratio, then we describe the constraints and stopping condition of the algorithm.

3.1.1 Entity ratio and Balancing

We have observed that NER datasets tend to hold a relatively stable ratio of entity tokens to total tokens. We refer to this ratio as b, and define it with respect to some labeled dataset as:

$$b = \frac{|P|}{|P| + |N|} \quad (2)$$

where N is the set of negative examples. Previous work has shown that in fully-annotated datasets the entity ratio tends to be about 0.09 ± 0.05, depending on the dataset and genre (Augenstein et al., 2017). Intuitively, knowledge of the gold entity ratio can help us estimate when we have found all the false negatives.

In our main experiments, we assume that the entity ratio with respect to the gold labeling is known for each training dataset. A similar assumption was made in Elkan and Noto (2008) when determining the c value, and in Grave (2014) in the constraint determining the percentage of OTHER examples. However, we also show in Section 4.8 that knowledge of this ratio is not strictly necessary, and a flat value across all datasets produces similar performance.

With a weighted training set, it is also useful to define the weighted entity ratio.

$$b = \frac{|P|}{|P| + \sum_{i \in N} v_i} \quad (3)$$

When training an NER model on weighted data, one can change the weighted entity ratio to achieve different effects. To make balanced predictions on test, the entity ratio in the training data should roughly match that of the test data (Chawla, 2005). To bias a model towards predicting positives or predicting negatives, the weighted entity ratio can be set higher or lower respectively. This effect is pronounced when using linear methods for NER, but not as clear in neural methods.

To change the entity ratio, we scale the weights in N by a scaling constant γ. Targeting a particular b^*, we may write:

$$b^* = \frac{|P|}{|P| + \gamma \sum_{i \in N} v_i} \quad (4)$$

We can solve for γ:

$$\gamma = \frac{(1 - b^*)|P|}{b^* \sum_{i \in N} v_i} \quad (5)$$

To obtain weights, v_i^*, that attain the desired entity ratio, b^*, we scale all weights in N by γ.

$$v_i^* = \gamma v_i \quad (6)$$

In the train-predict-infer loop, we balance the weights to a value near the gold ratio before training.

3.1.2 Constraints and Stopping Condition

We encode our constraints with an Integer Linear Program (ILP), shown in Figure 3. Intuitively, the job of the inference step is to take predictions ($\hat{T}$) and use knowledge of the task to 'fix' them.

In the objective function (Eqn. 8), token i is represented by two indicator variables y_{0i} and y_{1i}, representing negative and positive labels, respectively. Associated prediction scores C_0 and C_1 are from the classifier λ in the last round of predictions. The first constraint (Eqn. 9) encodes the fact that an instance cannot be both an entity and a non-entity.

The second constraint (Eqn. 10) enforces the ratio of positive to total tokens in the corpus to match a required entity ratio. $|T|$ is the total number of tokens in the corpus. b is the required entity ratio, which increases at each iteration. δ allows some flexibility, but is small.

Constraint 11 encodes that instances in P should be labeled positive since they were manually labeled and are by definition trustworthy. We set $\xi \geq 0.99$.

This framework is flexible in that more complex language- or task-specific constraints could be added. For example, in English and many other languages with Latin script, it may help to add a capitalization constraint. In languages with rich morphology, certain suffixes may indicate or contraindicate a named entity. For simplicity, and because of the number of languages in our experiments, we use only a few constraints.

After the ILP has selected predictions, we assign weights to each instance in preparation for training the next round. The decision process for an instance is:

$$v_i = \begin{cases} 1.0 & \text{If ILP labeled } x_i \text{ positive} \\ P_\lambda(y_i = \text{O} \mid x_i) & \text{Otherwise} \end{cases} \tag{7}$$

This is similar to Equation (1), except that the set of tokens that the ILP labeled as positive is larger than P. With new labels and weights, we start the next iteration.

The stopping condition for the algorithm is related to the entity ratio. One important constraint (Eqn. 10) governs how many positives are labeled at each round. This number starts at $|P|$ and is increased by a small value[3] at each iteration, thereby improving recall. Positive instances are chosen in two ways. First, all instances in P are constrained to be labeled positive (Eqn. 11). Second, the objective function ensures that high-confidence positives will be chosen. The stopping condition is met when the number of required positive instances (computed using gold unweighted entity ratio) equals the number of predicted positive instances.

4 Experiments

We measure the performance of our method on 8 different languages using artificially perturbed labels to simulate the partial annotation setting.

$$\max_{\mathbf{y}} \quad \sum_{i}^{|T|} C_{0i} y_{0i} + C_{1i} y_{1i} \tag{8}$$

$$\text{s.t.} \quad \forall i,\ y_{0i} + y_{1i} = 1 \tag{9}$$

$$b - \delta \leq \sum_i y_{1i}/|T| \leq b + \delta \tag{10}$$

$$\forall i,\ x_i \in P,\ \sum_i y_{1i} \geq \xi |P|, \tag{11}$$

Figure 3: ILP for the inference step

4.1 Data

We experiment on 8 languages. Four languages – English, German, Spanish, Dutch – come from the CoNLL 2002/2003 shared tasks (Tjong Kim Sang and De Meulder, 2003a,b). These are taken from newswire text, and have labelset of Person, Organization, Location, Miscellaneous.

The remaining four languages come from the LORELEI project (Strassel and Tracey, 2016). These languages are: Amharic (amh: LDC2016E87), Arabic (ara: LDC2016E89), Hindi (hin: LDC2017E62), and Somali (som: LDC2016E91). These come from a variety of sources including discussion forums, newswire, and social media. The labelset is Person, Organization, Location, Geo-political entity. We define train/development/test splits, taking care to keep a similar distribution of genres in each split. Data statistics for all languages are shown in Table 1.

4.2 Artificial Perturbation

We create partial annotations by perturbing gold annotated data in two ways: lowering recall (to simulate missing entities), and lowering precision (to simulate noisy annotations).

To lower recall, we replace gold named entity tags with O tags (for non-name). We do this by grouping named entity surface forms, and replacing tags on all occurrences of a randomly selected surface form until the desired amount remains. For example, if the token 'Bangor' is chosen to be untagged, then every occurrence of 'Bangor' will be untagged. We chose this slightly complicated method because the simplest idea (remove mentions randomly) leaves an artificially large diversity of surface forms, which makes the problem of discovering noisy entities easier.

To lower precision, we tag a random span (of a random start position, and a random length between 1 and 3) with a random named entity tag. We continue this process until we reach the desired precision. When both precision and recall are to be perturbed, the recall adjustment is made first, and then the number of random spans to be added is calculated by the entities that are left.

[3]The size of this value is related to how much we trust the ranking induced by prediction confidences. If we believed the ranking was perfect, we could take as many positives as we wanted and be finished in one round.

	Train			Test		
Lang.	b (%)	Tag	Tok	b (%)	Tag	Tok
English	16.6	23K	203K	17.3	5K	46K
Spanish	12.3	18K	264K	11.9	3K	51K
German	8.0	11K	206K	9.9	3K	51K
Dutch	9.5	13K	202K	8.3	4K	68K
Amharic	11.2	3K	52K	11.3	1K	18K
Arabic	12.6	4K	60K	10.2	931	16K
Hindi	7.38	4K	74K	7.53	1K	25K
Somali	11.2	4K	57K	11.9	1K	16K

Table 1: Data statistics for all languages, showing number of tags and tokens in Train and Test. The tag counts represent individual spans, not tokens. That is, "[Barack Obama]$_{\text{PER}}$" counts as one tag, not two. The b column shows the entity ratio as a percentage.

4.3 NER Models

In principle, CBL can use any NER method that can be trained with instance weights. We experiment with both non-neural and neural models.

4.3.1 Non-neural Model

For our non-neural system, we use a version of Cogcomp NER (Ratinov and Roth, 2009; Khashabi et al., 2018) modified to use Weighted Averaged Perceptron. This operates on a weighted training set $D_w = \{(x_i, y_i, v_i)\}_{i=1}^{N}$, where N is the number of training examples, and $v_i \geq 0$ is the weight on the ith training example. In this non-neural system, a training example is a word with context encoded in the features. We change only the update rule, where the learning rate α is multiplied by the weight:

$$\mathbf{w} = \mathbf{w} + \alpha v_i y_i (\mathbf{w}^T x_i) \tag{12}$$

We use a standard set of features, as documented in Ratinov and Roth (2009). In order to keep the language-specific resources to a minimum, we did not use any gazetteers for any language.[4] One of the most important features is Brown clusters, trained for 100, 500, and 1000 clusters for the CoNLL languages, and 2000 clusters for the remaining languages. We trained these clusters on Wikipedia text for the four CoNLL languages, and on the same monolingual text used to train the word vectors (described in Section 4.3.2).

4.3.2 Neural Model

A common neural model for NER is the BiLSTM-CRF model (Ma and Hovy, 2016). However, because the Conditional Random Field (CRF) layer calculates loss at the sentence level, we need a different method to incorporate token weights. We use a variant of the CRF that allows partial annotations by marginalizing over all possible sequences (Tsuboi et al., 2008).

[4]Separate experiments show that omitting gazetteers impacts performance only slightly.

When using a standard BiLSTM-CRF model, the loss of a dataset (D) composed of sentences (s) is calculated as:

$$\mathcal{L} = -\sum_{s \in D} \log P_\theta(\mathbf{y}^{(s)} | \mathbf{x}^{(s)}) \tag{13}$$

Where $P_\theta(\mathbf{y}^{(s)} | \mathbf{x}^{(s)})$ is calculated by the CRF over outputs from the BiLSTM. In the marginal CRF framework, it is assumed that $\mathbf{y}^{(s)}$ is necessarily partial, denoted as $\mathbf{y}_p^{(s)}$. To incorporate partial annotations, the loss is calculated by marginalizing over all possible sequences consistent with the partial annotations, denoted as $C(\mathbf{y}_p^s)$.

$$\mathcal{L} = -\sum_{s \in D} \log \sum_{\mathbf{y} \in C(\mathbf{y}_p^{(s)})} P_\theta(\mathbf{y} | \mathbf{x}^{(s)}) \tag{14}$$

However, this formulation assumes that all possible sequences are equally likely. To address this, Jie et al. (2019) introduced a way to weigh sequences.

$$\mathcal{L} = -\sum_{s \in D} \log \sum_{\mathbf{y} \in C(\mathbf{y}_p^{(s)})} q(\mathbf{y} | \mathbf{x}^{(s)}) P_\theta(\mathbf{y} | \mathbf{x}^{(s)}) \tag{15}$$

It's easy to see that this formulation is a generalization of the standard CRF if $q(.) = 1$ for the gold sequence $\mathbf{y}$, and 0 for all others.

The product inside the summation depends on tag transition probabilities and tag emission probabilities, as well as token-level "weights" over the tagset. These weights can be seen as defining a soft gold labeling for each token, corresponding to confidence in each label.

For clarity, define the soft gold labeling over each token x_i as $\mathbf{G}_i \in [0, 1]^L$, where L is the size of the labelset. Now, we may define $q(.)$ as:

$$q(\mathbf{y} | \mathbf{x}^{(s)}) = \prod_i G_i^{y_i}$$

Where $G_i^{y_i}$ is understood as the weight in $\mathbf{G}_i$ that corresponds to the label y_i.

We incorporate our instance weights in this model with the following intuitions. Recall that if an instance weight $v_i = 0$, this indicates low confidence in the label on token x_i, and therefore the labeling should not update the model at training time. Conversely, if $v_i = 1$, then this label is to be trusted entirely.

If $v_i = 0$, we set the soft labeling weights over x_i to be uniform, which is as good as no information. Since v_i is defined as confidence in the O label, the soft labeling weight for O increases proportionally to v_i. Any remaining probability mass is distributed evenly among the other labels.

To be precise, for tokens in N, we calculate values for $\mathbf{G}_i$ as follows:

$$G_i^O = \max(1/L, v_i)$$

$$G_i^{\text{non-}O} = \frac{1 - G_i^O}{L - 1}$$

For example, consider phase 1 of Constrained Binary Learning, in which the labelset is collapsed to two labels ($L = 2$). Assuming that the O label has index 0, then if $v_i = 0$, then $\mathbf{G}_i = [0.5, 0.5]$. If $v_i = 0.6$, then $\mathbf{G}_i = [0.6, 0.4]$.

For tokens in P (which have some entity label with high confidence), we always set $\mathbf{G}_i$ with 1 in the given label index, and 0 elsewhere.

We use pretrained GloVe (Pennington et al., 2014) word vectors for English, and the same pretrained vectors used in Lample et al. (2016) for Dutch, German, and Spanish. The other languages are distributed with monolingual text (Strassel and Tracey, 2016), which we used to train our own skip-n-gram vectors.

4.4 Baselines

We compare against several baselines, including two from prior work.

4.4.1 Raw annotations

The simplest baseline is to do nothing to the partially annotated data and train on it as is.

4.4.2 Instance Weights

Although CBL works with no initialization (that is, all tokens with weight 1), we found that a good weighting scheme can boost performance for certain models. We design weighting schemes that give instances in N weights corresponding to an estimate of the label confidence.[5] For example, non-name tokens such as *respectfully* should have weight 1, but possible names, such as *Russell*, should have a low weight, or 0. We propose two weighting schemes: frequency-based and window-based.

For the frequency-based weighting scheme, we observed that names have relatively low frequency (for example, *Kennebunkport*, *Dushanbe*) and common words are rarely names (for example *the*, *and*, *so*). We weigh each instance in N according to its frequency.

$$v_i^{\text{freq}} = freq(x_i) \tag{16}$$

where $freq(x_i)$ is the frequency of the i^{th} token in N divided by the count of the most frequent token. In our experiments, we computed frequencies over $P+N$, but these could be estimated on any sufficiently large corpus. We found that the neural model performed poorly when the weights followed a Zipfian distribution (e.g. most weights very small), so for those experiments, we took the log of the token count before normalizing.

For the window-based weighting scheme, noting that names rarely appear immediately adjacent to each other in English text, we set weights for tokens within a window of size 1 of a name (identified in P) to be 1.0, and for tokens farther away to be 0.

$$v_i^{\text{window}} = \begin{cases} 1.0 & \text{if } d_i \leq 1 \\ 0.0 & \text{otherwise} \end{cases} \tag{17}$$

where d_i is the distance of the i^{th} token to the nearest named entity in P.

Finally, we combine the two weighting schemes as:

$$v_i^{\text{combined}} = \begin{cases} 1.0 & \text{if } d_i \leq 1 \\ v_i^{\text{freq}} & \text{otherwise} \end{cases} \tag{18}$$

4.4.3 Self-training with Marginal CRF

Jie et al. (2019) propose a model based on marginal CRF (Tsuboi et al., 2008) (described in Section 4.3.2). They follow a self-training framework with cross-validation, using the trained model over all but one fold to update gold labeling distributions in the final fold. This process continues until convergence. They use a partial-CRF framework similar to ours, but taking predictions at face value, without constraints.

4.4.4 Neural Network with Noise Adaptation

Following Hedderich and Klakow (2018), we used a neural network with a noise adaptation layer.[6] This extra layer attempts to correct noisy examples given a probabilistic confusion matrix of label noise. Since this method needs a small amount of labeled data, we selected 500 random tokens to be the gold training set, in addition to the partial annotations.

As with our BiLSTM experiments, we use pretrained GloVe word vectors for English, and the same pretrained vectors used in Lample et al. (2016) for Dutch, German, and Spanish. We omit results from the remaining languages because the scores were substantially worse even than training on raw annotations.

4.5 Experimental Setup and Results

We show results from our experiments in Table 2. In all experiments, the training data is perturbed at 90% precision and 50% recall. These parameters are similar to the scores obtained by human annotators in a foreign language (see Section 5). We evaluate each experiment with both non-neural and neural methods.

First, to get an idea of the difficulty of NER in each language, we report scores from models trained on gold data without perturbation (*Gold*). Then we report results from an Oracle Weighting scheme (*Oracle Weighting*) that takes partially annotated data and assigns weights with knowledge of the true labels. Specifically, mislabeled entities in set N are given weight 0, and all other tokens are given weight 1.0. This scheme is free from labeling noise, but should still get lower scores than Gold because of the smaller number of entities. Since our method estimates these weights, we do not expect CBL to outperform the Oracle method. Next, we show results from all baselines. The bottom two sections are our results, first with no initialization (*Raw*), and CBL over that, then with *Combined Weighting* initialization, and CBL over that.

[5] All elements of P always have weight 1

[6] The code was kindly provided by the authors.

Method \ Language	Tool	**eng**	**deu**	**esp**	**ned**	**amh**	**ara**	**hin**	**som**	avg
Gold	Cogcomp	89.1	72.5	82.5	82.6	67.2	53.4	74.4	80.3	75.3
	BiLSTM-CRF	90.3	77.3	85.2	81.1	69.2	52.8	73.8	82.3	76.5
Oracle Weighting	Cogcomp	83.7	65.7	76.2	76.4	54.3	42.0	56.3	68.5	65.4
	BiLSTM-CRF	87.8	70.2	78.5	70.4	60.4	43.4	57.6	73.2	67.7
Noise Adaptation (Hedderich, 2018)		61.5	46.1	57.3	41.5	–	–	–	–	–
Self-training (Jie et al., 2019)		82.3	65.2	76.3	65.5	52.1	40.1	55.1	65.3	62.7
Raw Annotations	Cogcomp	54.8	36.9	49.5	47.9	31.0	32.6	30.9	44.0	40.9
	BiLSTM-CRF	73.3	57.7	61.9	58.3	42.2	36.8	47.5	54.9	54.1
CBL-Raw	CogComp	74.7	63.0	68.7	67.0	45.0	37.8	50.6	67.9	59.3
	BiLSTM-CRF	**84.6**	**67.9**	**79.6**	70.0	**52.9**	42.1	55.2	**70.4**	**65.3**
Combined Weighting	Cogcomp	75.2	56.6	70.8	70.8	46.5	**44.1**	57.5	60.2	60.2
	BiLSTM-CRF	73.5	60.3	64.9	61.9	48.0	38.0	49.0	56.6	56.5
CBL-Combined	Cogcomp	77.3	61.8	74.0	**72.4**	49.2	43.7	**58.2**	67.6	63.0
	BiLSTM-CRF	81.1	64.9	74.9	63.4	52.2	39.8	52.0	67.0	61.9

Table 2: F1 scores on English, German, Spanish, Dutch, Amharic, Arabic, Hindi, and Somali. Each section shows performance of both Cogcomp (non-neural) and BiLSTM (neural) systems. *Gold* is using all available gold training data to train. *Oracle Weighting* uses full entity knowledge to set weights on N. The next section shows prior work, followed by our methods. The column to the farthest right shows the average score over all languages. Bold values are the highest per column. On average, our best results are found in the uninitialized (*Raw*) CBL from BiLSTM-CRF.

4.6 Analysis

Regardless of initialization or model, CBL improves over the baselines. Our best model, *CBL-Raw BiLSTM-CRF*, improves over the *Raw Annotations BiLSTM-CRF* baseline by 11.2 points F1, and the *Self-training* prior work by 2.6 points F1, showing that it is an effective way to address the problem of partial annotation. Further, the best CBL version for each model is within 3 points of the corresponding *Oracle* ceiling, suggesting that this weighting framework is nearly saturated.

The *Combined* weighting scheme is surprisingly effective for the non-neural model, which suggests that the intuition about frequency as distinction between names and non-names holds true. It gives modest improvement in the neural model. The *Self-training* method is effective, but is outperformed by our best CBL method, a difference we discuss in more detail in Section 4.7. The *Noise Adaptation* method outperforms the *Raw annotations Cogcomp* baseline in most cases, but does not reach the performance of the *Self-training* method, despite using some fully labeled data.

It is instructive to compare the neural and non-neural versions of each setup. The neural method is better overall, but is less able to learn from the knowledge-based initialization weights. In the non-neural method, the difference between *Raw* and *Combined* is nearly 20 points, but the difference in the neural model is less than 3 points. *Combined* versions of the non-neural method outperform the neural method on 3 languages: Dutch, Arabic, and Hindi. Further, in the neural method, *CBL-Raw* is always worse than *CBL-Combined*. This may be due to the way that weights are used in each model. In the non-neural model, a low enough weight completely cancels the token, whereas in the neural model it is still used in training. Since the neural model performs well in the *Oracle* setting, we know that it can learn from hard weights, but it may have trouble with the subtle differences encoded in frequencies. We leave it to future work to discover improved ways of incorporating instance weights in a BiLSTM-CRF.

In seeking to understand the details of the other results, we need to consider the precision/recall tradeoff. First, all scores in the *Gold* row had higher precision than recall. Then, training on raw partially annotated data biases a classifier strongly towards predicting few entities. All results from the *Raw annotations* row have precision more than double the recall (e.g. Dutch Precision, Recall, F1 were: 91.5, 32.4, 47.9). In this context, the problem this paper explores is how to improve the recall of these datasets without harming the precision.

4.7 Difference from Prior Work

While our method has several superficial similarities with prior work, most notably Jie et al. (2019), there are some crucial differences.

Our methods are similar in that they both use a model trained at each step to assign a soft gold-labeling to each token. Each algorithm iteratively trains models using weights from the previous steps.

One difference is that Jie et al. (2019) use cross-validation to train, while we follow Chang et al. (2007) and retrain with the entire training set at each round.

	Avg F1		
Method \ b	10%	15%	Gold
Oracle Weighting	65.8	**65.9**	65.4
Raw annotations	40.9	40.9	40.9
Combined Weighting	59.9	**60.2**	**60.2**
CBL-Combined	62.4	62.3	**63.0**

Table 3: Experimenting with different entity ratios. Scores reported are average F1 across all languages. *Gold* b value refers to using the gold annotated data to calculate the optimal entity ratio. This table shows that exact knowledge of the entity ratio is not required for CBL to succeed.

However, the main difference has to do with the focus of each algorithm. Recall the discussion in Section 3 regarding the two possible approaches of **1**) find the false negatives and label them correctly, and **2**) find the false negatives and remove them. Conceptually, the former was the approach taken by Jie et al. (2019), the latter was our approach. Another way to look at this is as focusing on predicting correct tag labels ((Jie et al., 2019)) or focus on predicting O tags with high confidence (ours).

Even though they use soft labeling (which they show to be consistently better than hard labeling), it is possible that the predicted tag distribution is incorrect. Our approach allows us to avoid much of the inevitable noise that comes from labelling with a weak model.

4.8 Varying the Entity Ratio

Recall that the entity ratio is used for balancing and for the stopping criteria in CBL. In all our experiments so far, we have used the gold entity ratio for each language, as shown in Table 1. However, exact knowledge of entity ratio is unlikely in the absence of gold data. Thus, we experimented with selecting a default b value, and using it across all languages, with the Cogcomp model. We chose values of 10% and 15%, and report F1 averaged across all languages in Table 3.

While the gold b value is the best for *CBL-Combined*, the flat 15% ratio is best for all other methods, showing that exact knowledge of the entity ratio is not necessary.

5 Bengali Case Study

So far our experiments have shown effectiveness on artificially perturbed labels, but one might argue that these systematic perturbations don't accurately simulate real-world noise. In this section, we show how our methods work in a real-world scenario, using Bengali data partially labeled by non-speakers.

5.1 Non-speaker Annotations

In order to compare with prior work, we used the train/test split from Zhang et al. (2016). We removed all gold labels from the train split, romanized it[7] (Hermjakob et al., 2018), and presented it to two non-Bengali speaking annotators using the TALEN interface (Mayhew and Roth, 2018). The instructions were to move quickly and annotate names only when there is high confidence (e.g. when you can also identify the English version of the name). They spent about 5 total hours annotating, without using Google Translate. This sort of non-speaker annotation is possible because the text contains many 'easy' entities – foreign names – which are noticeably distinct from native Bengali words. For example, consider the following:

Num tokens	49K
Num sentences	2435
Num name tokens	2326
Entity ratio	4.66%
Num unique name tokens	664
Annotator 1 Prec/Rec/F1	84/34/48
Annotator 2 Prec/Rec/F1	79/28/42
Combined Prec/Rec/F1	83/32/47

Table 4: Bengali Data Statistics. The P/R/F1 scores are computed for the non-speaker annotator with respect to the gold training data.

- **Romanized Bengali**: ebisi'ra giliyyaana phinnddale aaja pyaalestaaina adhiinastha gaajaa theke aaja raate ekhabara jaaniyyechhena .
- **Translation**[8]: ABC's Gillian Fondley has reported today from Gaza under Palestine today.

The entities are Gillian Findlay, ABC, Palestine, and Gaza. While a fast-moving annotator may not catch most of these, 'pyaalestaaina' could be considered an 'easy' entity, because of its visual and aural similarity to 'Palestine.' A clever annotator may also infer that if Palestine is mentioned, then Gaza may be present.

Annotators are moving fast and being intentionally non-thorough, so the recall will be low. Since they do not speak Bengali, there are likely to be some mistakes, so the precision may drop slightly also. This is exactly the noisy partial annotation scenario addressed in this paper. The statistics of this data can be seen in Table 4, including annotation scores computed with respect to the gold training data for each annotator, as well as the combined score.

We show results in Table 5, using the BiLSTM-CRF model. We compare against other low-resource approaches published on this dataset, including two based on Wikipedia (Tsai et al., 2016; Pan et al., 2017), another based on lexicon translation from a high-resource language (Mayhew et al., 2017). These prior methods operate under somewhat different paradigms than

[7]This step is vitally important. We used `www.isi.edu/~ulf/uroman.html`

[8]From `translate.google.com`

	Test		
Scheme	P	R	F1
(Zhang et al., 2016)	-	-	34.8
(Tsai et al., 2016)	-	-	43.3
(Pan et al., 2017)	-	-	44.0
(Mayhew et al., 2017)	-	-	46.2
BILSTM-CRF			
Train on Gold	71.6	70.2	70.9
Raw annotations	**73.0**	23.8	35.9
Combined Weighting	65.9	34.2	45.0
CBL-Raw	57.8	**47.3**	**52.0**
CBL-Combined	58.3	44.2	50.2

Table 5: Bengali manual annotation results. Our methods improve on state of the art scores by over 5 points F1 given a relatively small amount of noisy and incomplete annotations from non-speakers.

this work, but have the same goal: maximizing performance in the absence of gold training data.

Raw annotations is defined as before, and gives similar high-precision low-recall results. The *Combined Weighting* scheme improves over Raw annotations by 10 points, achieving a score comparable to the prior state of the art. Beyond that, *CBL-Raw* outperforms the prior best by nearly 6 points F1, although *CBL-Combined* again underwhelms.

To the best of our knowledge, this is the first result showing a method for non-speaker annotations to produce high-quality NER scores. The simplicity of this method and the small time investment for these results gives us confidence that this method can be effective for many low-resource languages.

6 Conclusions

We explore an understudied data scenario, and introduce a new constrained iterative algorithm to solve it. This algorithm performs well in experimental trials in several languages, on both artificially perturbed data, and in a truly low-resource situation.

7 Acknowledgements

This work was supported by Contracts HR0011-15-C-0113 and HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Generalisation in named entity recognition: A quantitative analysis. *Computer Speech and Language*, 44:61–83.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 280–287, Prague, Czech Republic. Association for Computational Linguistics.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Nitesh V. Chawla. 2005. Data mining for imbalanced datasets: An overview. In *The Data Mining and Knowledge Discovery Handbook*.

Mostafa Dehghani, Arash Mehrjou, Stephan Gouws, Jaap Kamps, and Bernhard Schölkopf. 2017. Fidelity-weighted learning. *CoRR*, abs/1711.02799.

Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM.

Eraldo Rezende Fernandes and Ulf Brefeld. 2011. Learning from partially annotated sequences. In *ECML/PKDD*.

Daniel Flannery, Yusuke Miyao, Graham Neubig, and Shinsuke Mori. 2012. A pointwise approach to training dependency parsers from partially annotated corpora. *Information and Media Technologies*, 7(4):1489–1513.

Jason A. Fries, Sen Wu, Alexander Ratner, and Christopher Ré. 2017. Swellshark: A generative model for biomedical named entity recognition without labeled data. *CoRR*, abs/1704.06360.

Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *ICLR*.

Edouard Grave. 2014. Weakly supervised named entity classification. In *AKBC*.

Michael A. Hedderich and Dietrich Klakow. 2018. Training a neural network in a low-resource setting on automatically annotated noisy data. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 12–18, Melbourne. Association for Computational Linguistics.

Ulf Hermjakob, Jonathan May, and Kevin Knight. 2018. Out-of-the-box universal Romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, Melbourne, Australia. Association for Computational Linguistics.

Jerónimo Hernández-González, Inaki Inza, and Jose A Lozano. 2016. Weak supervision and other non-standard classification problems: a taxonomy. *Pattern Recognition Letters*, 69:49–55.

Dirk Hovy and Eduard Hovy. 2012. Exploiting partial annotations with EM training. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 31–38, Montréal, Canada. Association for Computational Linguistics.

Zhanming Jie, Pengjun Xie, Wei Lu, Ruixue Ding, and Linlin Li. 2019. Better modeling of incomplete annotations for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 729–734, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniel Khashabi, Mark Sammons, Ben Zhou, Tom Redman, Christos Christodoulopoulos, Vivek Srikumar, Nicholas Rizzolo, Lev Ratinov, Guanheng Luo, Quang Do, Chen-Tse Tsai, Subhro Roy, Stephen Mayhew, Zhili Feng, John Wieting, Xiaodong Yu, Yangqiu Song, Shashank Gupta, Shyam Upadhyay, Naveen Arivazhagan, Qiang Ning, Shaoshi Ling, and Dan Roth. 2018. Cogcompnlp: Your swiss army knife for nlp. In *LREC*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Wee Sun Lee and Bing Liu. 2003. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*.

Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. 2003. Building text classifiers using positive and unlabeled examples. In *ICDM*.

Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. 2002. Partially supervised classification of text documents. In *ICML*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Stephen Mayhew and Dan Roth. 2018. TALEN: Tool for annotation of low-resource ENtities. In *Proceedings of ACL 2018, System Demonstrations*, pages 80–86, Melbourne, Australia. Association for Computational Linguistics.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shinsuke Mori, Yosuke Nakata, Graham Neubig, and Tetsuro Sasada. 2015. Pointwise prediction and sequence-based reranking for adaptable part-of-speech tagging. In *PACLING*.

Qiang Ning, Zhongzhi Yu, Chuchu Fan, and Dan Roth. 2018. Exploiting partially annotated data in temporal relation extraction. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 148–153, New Orleans, Louisiana. Association for Computational Linguistics.

Joel Nothman, James R Curran, and Tara Murphy. 2008. Transforming wikipedia into named entity training data. In *Proceedings of the Australian Language Technology Workshop*, pages 124–132.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artif. Intell.*, 194:151–175.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*.

Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, 11 3:269–282.

Dan Roth. 2017. Incidental supervision: Moving beyond supervised learning. In *Proc. of the Conference on Artificial Intelligence (AAAI)*.

Stephanie Strassel and Jennifer Tracey. 2016. LORELEI language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3273–3280, Portorož, Slovenia. European Language Resources Association (ELRA).

Erik F. Tjong Kim Sang and Fien De Meulder. 2003a. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003b. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*.

Yuta Tsuboi, Hisashi Kashima, Hiroki Oda, Shinsuke Mori, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 897–904. Association for Computational Linguistics.

Boliang Zhang, Xiaoman Pan, Tianlu Wang, Ashish Vaswani, Heng Ji, Kevin Knight, and Daniel Marcu. 2016. Name tagging for low-resource incident languages based on expectation-driven learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 249–259, San Diego, California. Association for Computational Linguistics.

Contextualized Cross-Lingual Event Trigger Extraction with Minimal Resources

Meryem M'hamdi , Marjorie Freedman and Jonathan May
Information Sciences Institute & Computer Science Department
University of Southern California (USC)
{meryem, mrf, jonmay}@isi.edu

Abstract

Event trigger extraction is an information extraction task of practical utility, yet it is challenging due to the difficulty of disambiguating word sense meaning. Previous approaches rely extensively on hand-crafted language-specific features and are applied mainly to English for which annotated datasets and Natural Language Processing (NLP) tools are available. However, the availability of such resources varies from one language to another. Recently, contextualized Bidirectional Encoder Representations from Transformers (BERT) models have established state-of-the-art performance for a variety of NLP tasks. However, there has not been much effort in exploring language transfer using BERT for event extraction.

In this work, we treat event trigger extraction as a sequence tagging problem and propose a cross-lingual framework for training it without any hand-crafted features. We experiment with different flavors of transfer learning from high-resourced to low-resourced languages and compare the performance of different multilingual embeddings for event trigger extraction. Our results show that training in a multilingual setting outperforms language-specific models for both English and Chinese. Our work is the first to experiment with two event architecture variants in a cross-lingual setting, to show the effectiveness of contextualized embeddings obtained using BERT, and to explore and analyze its performance on Arabic.

1 Introduction

Event trigger extraction, as defined the Automatic Content Extraction multilingual evaluation benchmark (ACE2005) (Walker, 2006), is a subtask of event extraction which requires systems to detect and label the lexical instantiation of an event, known as a *trigger*. As an example, in the sentence "John **traveled** to NYC for a meeting", **traveled** is a trigger of a *Movement-Transport* event. Trigger detection is typically the first step in extracting the structured information about an event (e.g. the time, place, and participant arguments; distinguishing between past, habitual, and future events). This definition of the task restricts it to events that can be triggered explicitly by actual words and makes it context-vulnerable: the same event might be expressed by different triggers and a specific trigger can represent different event types depending on the context.

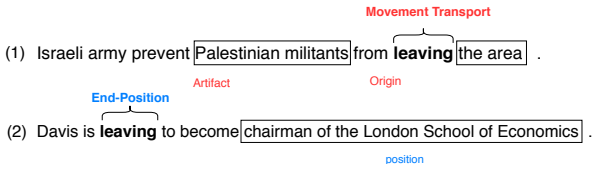


Figure 1: Examples of importance of context in trigger disambiguation.

Event trigger extraction is challenging as it involves understanding the context in order to be able to identify the event that the trigger refers to. Figure 1 shows two examples where context plays a crucial role in disambiguating the word sense of **leaving**, which is a trigger for a *Movement-Transport* event in the first sentence and for an *End-Position* event in the second sentence.

Due to the complexity of the task and the difficulty in constructing a standard annotation scheme, there exists limited labeled data, for only a few languages. The earliest work has focused mainly on English, for which there are relatively many annotated sentences, and relies extensively on language-specific linguistic tools to extract the lexical and syntactic features that need to be computed as a pre-requisite for the task (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Li et al., 2013).

Simply generating annotated corpora for each

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 656–665
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

language of interest is not only costly and time-consuming, it is also not necessarily guaranteed to address the **great NLP divide**, where performance depends on the language, the ability to generate language-specific features, and the quality tools (in this case, syntactic parsers) available for each language. In an attempt to reduce the great NLP divide, we observe a tendency of practitioners drifting away from linguistic features and more towards continuous distributed features that can be obtained without hand-engineering, based simply on publicly available corpora. Recently, approaches have tried to overcome the limitation of traditional lexical features, which can suffer from the data sparsity problem and inability to fully capture the semantics of the words, by making use of sequential modeling methods including variants of Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), and/or Conditional Random Fields (CRF). (Chen et al., 2015; Liu et al., 2016; Nguyen et al., 2016; Sha et al., 2018; Liu et al., 2018b).

Existing approaches which take into consideration the cross-lingual aspect of event trigger extraction tend to either take inspiration from machine translation, distant supervision or multitasking. Machine translation is used by Liu et al. (2018a) to project monolingual text to parallel multilingual texts to model the confidence of clues provided by other languages. However, this approach suffers from error propagation of machine translation.

Another approach relies on multilingual embeddings, which can be pre-trained beforehand on large monolingual corpora, using no explicit parallel data, and bridging the gap between different languages by learning a way to align them into a shared vector space. The ability of these models to represent a common representation of words across languages makes them attractive to numerous downstream NLP applications. Multilingual Unsupervised and Supervised Embeddings (MUSE) is a framework for training cross-lingual embeddings in an unsupervised manner, which leads to competitive results, even compared to supervised approaches (Conneau et al., 2017). However, there is no prior work leveraging this kind of representation for cross-lingual event trigger extraction.

More recently, BERT, a deep bidirectional representation which jointly conditions on both left and right context (Devlin et al., 2019), was proposed, which unlike MUSE, provides *contextualized* word embeddings, and has been shown to achieve state-of-the-art performance on many NLP tasks. In particular, (Yang et al., 2019) propose a method based on BERT for enhancing event trigger and argument extraction by generating more labeled data. However, it has not been applied in the context of cross-lingual transfer learning.

In this paper, we investigate the possibility of automatically learning effective features from data while relying on zero language-specific linguistic resources. Moreover, we explore the application of multilingual embeddings to the event trigger extraction task in a **direct transfer of annotation scheme** where ground truth is only needed for one language and can be used to predict labels in other languages and **other boosted and joint multilingual schemes**. We perform a systematic comparison between training using monolingual versus multilingual embeddings and the difference in gain on performance with respect to different train/test language pairs. We evaluate our framework using two embedding approaches: type-based unsupervised embeddings (MUSE) and contextualized embeddings (BERT). For the latter, we demonstrate that our proposed model achieves a better (or comparable) performance for all languages compared to some benchmarks for event extraction on the ACE2005 dataset.

The main contributions of the paper can be summarized as follows:

(1) We apply different state-of-the-art NN architectures for sequence tagging on trigger extraction and compare them to feature-based baselines and multilingual projection based models.

(2) We achieve a better performance using contextualized word representation learning in event trigger extraction backed up with both quantitative and qualitative analysis.

(3) We evaluate the effectiveness of a multilingual approach using zero-shot transfer learning, targeted cross-lingual and joint multilingual training schemes.

(4) We investigate event trigger extraction performance when using Arabic.

2 Methodology

We treat trigger extraction as a sequence tagging problem for which we start by designing a ba-

sic state-of-the-art approach for sequence tagging based on bidirectional Long Short Term Memory (bi-LSTM) with word and character embeddings and a CRF layer on top of it. Then, we describe an approach that trains BERT with a CRF layer for the task. In both architectures, the input is in the form of BIO notation used to differentiate between the beginning (B), inside (I) and (O) for no associated trigger labels.

2.1 Bi-LSTM-Char-CRF networks

The Bi-LSTM-Char-CRF for sequence tagging model is a hierarchical neural network model based on three components: character-level using character embeddings, word-level using bi-LSTM over word embeddings and sequence-level using CRF. The architecture of the model is depicted in Figure 2.

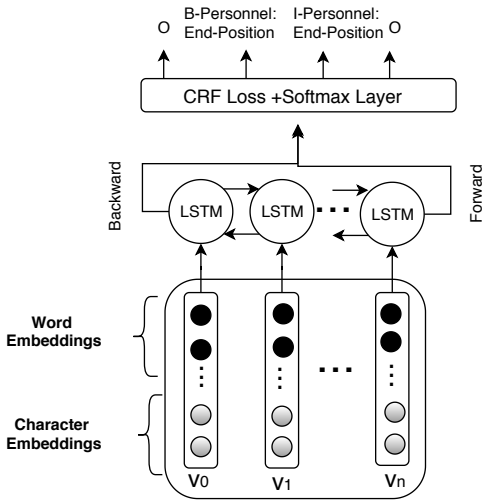


Figure 2: Bidirectional LSTM with character embeddings and CRF layer

2.1.1 Bi-LSTM networks

LSTMs (Hochreiter and Schmidhuber, 1997) are variants of RNNs that help learn long-range dependencies efficiently thanks to their use of memory and forget cells. Those cells help control the amount of the input to be retained/forgotten from previous states.

Given an input character or word embeddings representation x_t for a given time step t, we use bidirectional LSTMs by encoding features in their forward: $fh_i = \overrightarrow{LSTM}(x_i)$ and backward $bh_i = \overleftarrow{LSTM}(x_i)$ directions and concatenating them $h_i = [fh_i, bh_i]$ to capture information from both the past and future.

2.1.2 Character Embeddings

Character embeddings are used to capture orthographic patterns and to deal with out-of-vocabulary words, especially in the cross-lingual setting. We follow the same setup as Lample et al. (2016) to obtain character embeddings using bi-LSTM. Specifically, we concatenate both character and word-level features and use a bi-LSTM on top of that.

2.1.3 CRF Layer

The encoded character and word-level features are fed to a CRF layer to learn inter-dependencies between output trigger tags and find the optimal tag sequence. This layer simulates bi-LSTM in its use of past and future tags to predict the current tag. Following Lafferty et al. (2001), CRF layers define a transition matrix A and use a score A_{ij} to model the transition from the i^{th} state to the j^{th} for a pair of consecutive time steps. The scores $[f_\theta]_{i,t}$ of the matrix is the score output by the network with parameters θ, for the sentence $[x]_1^N$ and for the i^th tag, at the t^th word. The score of a sequence of tags $[y]_1^N$ for a particular sequence of words $[x]_1^N$ is the sum of transition scores and network scores which are computed efficiently using dynamic programming.

$$s([x]_1^N, [i]_1^N) = \sum_{t=1}^{N} ([A]_{[i]_{t-1},[i]_t} + [f_\theta]_{[i]_t,t}) \quad (1)$$

2.2 BERT-CRF

BERT is a multi-layer bidirectional transformer encoder, an extension to the original Transformer model (Vaswani et al., 2017). The input representation consists of a concatenation of WordPiece embeddings (Wu et al., 2016), positional embeddings, and the segment embedding. A special token ([CLS]) is inserted at the beginning of each sentence and another special token ([PAD]) is used to normalize the length of sentences (no ([SEP]) token is used in this case). The pre-trained BERT model provides a powerful contextualized representation which gives the state-of-the-art performance for many NLP tasks. We use BERT-CRF, which adds a CRF layer on top of BERT's contextualized embeddings layer.

3 Experimental Setup

3.1 Dataset

We evaluate our approach on the ACE2005 sentence-level event mention multilingual bench-

mark.[1] This dataset is annotated with 33 event subtypes which, when represented in BIO annotation, results in a 67-way labeling task. For a sound comparison, we use the same data split as the English baseline (as detailed in Section 3.3). To the best of our knowledge, there are no Arabic benchmark systems, so we produced our own split.[2] Statistics of the split for train, validation, and test for the three languages: English (EN), Chinese (ZH) and Arabic (AR) are included in Table 1.

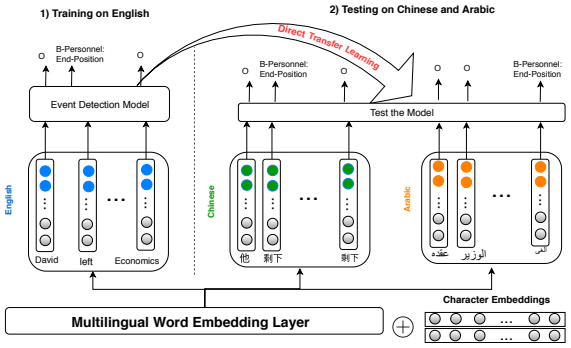


Figure 3: A zero-shot transfer learning architecture for cross-lingual event trigger extraction

3.2 Evaluation

We design different experiments for the evaluation of trigger extraction, where we train several language-specific and multilingual models using different embeddings and sequence labeling architectures. We evaluate the following training schemes:

- *Monolingual Baselines*: We train and fine-tune on EN, ZH or AR using monolingual FastText embeddings and testing on the trained language.
- *Zero-Shot learning experiments*: As depicted in Figure 3, we train and fine-tune on EN using multilingual embeddings (MUSE or BERT(multi)) and test on ZH and AR assuming no resources for those languages. To simplify experiments, we evaluate direct transfer only from EN since it is a high-resourced target language for learning projections needed in multilingual embeddings. We also believe AR and ZH are not good language-pair candidates, so we expect training on AR and testing on ZH and EN or training on ZH and testing on AR and EN would not lead to improvements.
- *Targeted cross-lingual experiments*: For each test language (ZH and AR), we train and fine-tune using multilingual embeddings on language pairs involving the test language in addition to EN to test to what extent adding training instances from the target language boosts the performance over zero-shot learning from EN only. When testing on EN, we train on EN+AR and EN+ZH.
- *Joint multilingual experiments*: We train and fine-tune on all languages (EN, ZH, and AR) using multilingual embeddings and testing on EN, ZH and AR. The hypothesis to be tested is whether a single language-independent model can work well across languages.

3.3 Baselines

We compare our methodology against different systems based on:

- Discrete Only: hand-crafted features only; *Ji's Cross-Entity'08* (Ji and Grishman, 2008); *Liao's Cross-Event'10* (Liao and Grishman, 2010); and *Li's Joint-Beam'13* (Li et al., 2013).
- Discrete + Continuous: using a combination of both linguistic features and trainable continuous features; *Chen's Dynamic CNN (DMCNN'15)* (Chen et al., 2015); *Nguyen's Joint RNN (JRNN'16)* (Nguyen et al., 2016); *Liu's Jointly Multiple Events (JMEE'18)* (Liu et al., 2018b); and *Zhang's Generative Adversarial Imitation Learning (GAIL'19)* (Zhang and Ji, 2018).
- Continuous Only: language-independent features only; *Feng's Hybrid Neural Network (HNN)'16* (Feng et al., 2016) and *Liu's Gated Multilingual Attention (GMLATT)'18* (Liu et al., 2018a).

For cross-lingual results, we include a comparison with ZH baselines; *Li's Maximum Entropy (MaxEnt)'13* (Li et al., 2013); *Chen's Rich-C'12* (Chen and Ng, 2012); *Feng's HNN'16* (Feng et al.,

[1] https://catalog.ldc.upenn.edu/LDC2006T06

[2] Our document splits are available in: https://github.com/meryemmhamdi1/cross-ling-ev-extr

Lang	Training		Validation		Test	
	#doc	#triggers	#doc	#triggers	#doc	#triggers
EN	529	4,420	30	505	40	424
ZH	557	2,213	32	111	44	197
AR	354	1,986	21	112	28	169

Table 1: Number of training, dev, test triggers and documents per language in ACE2005 dataset

2016); and *Hsi's Multi'16* (Hsi et al., 2016). More decriptions are included in Sections 5.1 and 5.2.

3.4 Hyper-parameters and Embeddings

We describe the hyper-parameters leading to the best attainable performance for each event trigger extraction architecture. They are selected based on random search and performance on the validation dataset. For Bi-LSTM-Char-CRF, we train character embeddings using a single bi-LSTM layer with 100 hidden units and use another single layer of bi-LSTM with 300 hidden units to train on the concatenated word and char embeddings. We use a dropout rate of 0.5. We optimize using Adam with learning rate of 0.01, weight decay rate of 0.9, $\beta_1 = 0.7$, $\beta_2 = 0.999$ and $\epsilon = 1\text{e}{-8}$.

For monolingual embeddings, we use 300-dimensional word embeddings for EN, ZH, and AR from fastText (Bojanowski et al., 2017). For multilingual experiments, we use MUSE library [3] to train unsupervised alignments from ZH and AR to EN resulting in a unified vector space for the three languages. We use the same training hyper-parameters across monolingual and multilingual training to ensure a fair comparison.

For BERT-CRF, we train monolingual EN and ZH using cased BERT-Base and BERT-ZH models[4] respectively and for all multilingual experiments, we use the recommended multi-cased BERT-Base model.[5] All models were trained using 12 layers with 768 hidden size and 12 self-attention heads and 110 Million parameters. We fine tune all BERT models with their default parameters. We use Adam with learning rate of 0.01, weight decay rate of 0.9, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1\text{e}{-6}$.

For all experiments, we use a batch size of 32 and limit the maximum sequence length of sentences to 128, padding or cutting otherwise. In the end, we report F1 for both trigger identification and classification tasks computed using the seqeval[6] framework for sequence labeling evaluation based on the CoNLL-2000 shared task, complying with previous work. Trigger classification doesn't assume the identification is correct but rather gives a stricter performance metric for measuring whether the trigger is not only identified but also correctly classified.

4 Results

Table 2 shows F1 scores of trigger identification and classification tested on EN, ZH and AR across two event architectures: Bi-LSTM-Char-CRF and BERT-CRF and using different embeddings and training schemes (fine-grained performance analysis based on precision, recall scores can be found in Appendix A).

4.1 Comparison with Feature-Based State-of-the-art

Before digging deeper into the comparison of our results with previous state-of-the-art methodology, it is worth comparing the different approaches taken by the prior work. For both EN and ZH, we observe that the best F1 scores over trigger identification and classification are obtained by Liu's JMEE in the first place and Feng's HNN with a close performance (with scores of 73.7% and 73.4% on trigger classification). For the multilingual case (ZH), it is clear that Feng's HNN is very competitive, whereas models relying on machine translation, namely Liu's GMLATT and Hsi's multi, lag behind the rest of models.

It is not surprising that a neural-based system outperforms other hand-crafted architectures since the former can capture richer sequence information beyond sentence-level than traditional NLP pre-processing such as dependency parsing and avoid errors propagated from such tools.

[3] https://github.com/facebookresearch/MUSE

[4] No pre-trained BERT model exists for Arabic.

[5] https://github.com/google-research/bert

[6] https://github.com/chakki-works/seqeval

Model	Train Lang	Embed -dings	Test Lang					
			EN		ZH		AR	
			Ident	Class	Ident	Class	Ident	Class
† *Ji's Cross-Entity'08*	EN	-	N/A	68.3	-	-	-	-
† *Liao's Cross-Event'10*		-	N/A	68.8	-	-	-	-
† *Li's Joint-Beam'13*		-	70.4	67.5	-	-	-	-
‡ *Chen's DMCNN'15*		Skip-Gram	73.5	69.1	-	-	-	-
‡ *Nguyen's JRNN'16*		C-BOW	71.9	69.3	-	-	-	-
‡ *Lius's JMEE'18*		Glove	75.9	73.7	-	-	-	-
‡ *Zhang's GAIL'19*		ELMo	73.9	72.0	-	-	-	-
+ *Feng's HNN'16*		Skip-Gram	75.9	73.4	-	-	-	-
+ Liu's GMLATT'18		Skip-Gram	74.1	72.4	-	-	-	-
† Li's *MaxEnt'13*	ZH	-	-	-	60.6	57.6	-	-
† *Chen's Rich-C'12*		-	-	-	66.7	63.2	-	-
‡ *Hsi's Multi'16*		multi_proj	-	-	N/A	39.6	-	-
+ *Feng's HNN'16*		Skip-Gram	-	-	68.2	63.0	-	-
* **Bi-LSTM-Char-CRF**	Test Lang	FastText	67.5	63.2	86.6	69.5	54.9	52.8
	Test Lang	MUSE	68.9	62.5	29.6	25.0	20.3	18.7
	EN		-	-	61.3	48.8	53.0	42.2
	EN+ZH		69.5	65.8	77.2	70.6	-	-
	EN+AR		70.6	66.9	-	-	56.1	53.2
	All		66.5	61.6	72.6	64.3	69.4	62.3
* **BERT-CRF**	Test Lang	Bert(Base)	79.2	**75.3**	84.4	79.9	-	-
	Test Lang	BERT (multi)	77.8	73.1	83.7	79.8	69.8	66.7
	EN		-	-	76.8	68.5	37.8	30.9
	EN+ZH		**79.8**	75.2	84.7	81.2	-	-
	EN+AR		79.3	74.5	-	-	**74.9**	**69.5**
	All		79.2	73.5	**87.7**	**83.2**	73.2	68.9

Table 2: Comparison of performance with different train/test language pairs using prior work baselines in the 1^{st} half and our method using Bi-LSTM-Char-CRF and BERT-CRF in the 2^{nd} half. †, + and ‡ denote baseline approaches using Discrete Only, Discrete + Continuous and Continuous Only features respectively, whereas * denotes our own approaches. A '-' designates that the experiment doesn't apply for that test language.

We observe that in general our language-independent (monolingual) Bi-LSTM-Char-CRF and BERT-CRF methods are on par with or outperform best attainable results. In particular, **BERT-CRF trained monolingually using BERT(Base) embeddings outperforms other baselines** for both EN and ZH, with F1-scores of 79.2 and 75.3 on trigger identification and classification, respectively, amounting to a 3.3% and 1.6% gain for EN. For ZH, we obtain F1-scores of 84.4% and 79.9%, amounting to an increase of 16.2% and 16.9% over the previous state-of-the-art. On the other hand, although results using Bi-LSTM-Char-CRF lag behind state-of-the-art for EN, incurring a loss of 10.5% over trigger classification, they are competitive for ZH, with scores of 86.6% and 69.5% and gains of 17.9% and 6.2% over Feng's HNN for trigger identification and classification respectively.

4.2 Comparison between MUSE and BERT Embeddings

We observe a significant difference in performance **in favor of BERT-CRF compared to Bi-LSTM-Char-CRF with a gain of 12.1%, 10.4%, and 13.9%** on the classification task. The better performance of BERT-CRF compared to Bi-LSTM-Char-CRF can be attributed to the fact that BERT is able to learn contextualized representation and long-range dependencies at different levels of granularity. Table 3 provides some examples where the surface form of the trigger is hard to dis-

ambiguate without context information. Reconsidering the second example from the introduction, we notice that a Bi-LSTM-Char-CRF fails to effectively associate it with position context clues.

4.3 Cross-lingual Event Trigger Extraction

In general, we observe that multilingual training leveraging multilingual embeddings provides a boost in performance for both event architectures, especially for EN and AR. More precisely, there is a **gain of 3.1% and 4.0% for EN and ZH respectively** on the identification performance of multilingual over monolingual models. We notice that AR benefits the most from multilingual training with an **improvement of 9.5% and 2.8%** on the classification score with BERT-CRF and Bi-LSTM-Char-CRF respectively. This supports our claim about the effectiveness of multilingual models which are efficient to train and are more robust than monolingual models.

Although F1-scores for **zero-shot transfer learning**(train: EN, test: ZH/AR) are not the best among multilingual experiments, they are still promising and exceed prior published work given the fact that no data from the target language was used to fine-tune. In particular, **training with EN using BERT-CRF was helpful for ZH** with a performance not far from monolingual performance. The same can be observed in the case of EN→ZH and EN→AR using MUSE. The lower performance of EN→AR using BERT-CRF raises questions about the quality of BERT(multi) embedding model training for Arabic.

Not surprisingly, training given a reasonable amount of language-specific resources from the test language under a **targeted cross-lingual scheme**(train: EN+ZH/EN+AR, test: EN/ZH/AR), boosts (with rare exceptions) the performance over both monolingual training and zero-shot learning: EN+AR>EN, EN+ZH>ZH>EN and EN+AR>EN>AR when testing on ZH, AR and even for EN for which we have a strong monolingual baseline.

When all languages are used to train **one single joint multilingual model** (train: All, test: EN/AR/ZH) we don't always notice improvements over monolingual models. To gain more insight into why multilingual training boosts performance over monolingual models, we include some examples of when EN is complementary to ZH and AR and without which the model fails to identify some events. In the Chinese example, there are only 12 "nearby" Chinese words to the trigger word 解散 (Jiěsàn) in ZH training data, whereas there are 4 times as many nearby words in EN (e.g. disband, dissolve, shut, cease, etc).

5 Related Work

Since this work is at the intersection of (i) event extraction and (ii) multilingual event extraction, we present previous work in relation to each domain separately in addition to (iii) a description of cross-lingual approaches for other tasks which motivate our current work.

5.1 Event Extraction in English

Previous works in event extraction on ACE2005 benchmark dataset are mostly focused on English and can be categorized based on the degree of hand-crafted features used and whether they are trained in a pipelined or joint fashion. While some systems such as Cross-Document (Ji and Grishman, 2008) and Cross-Event (Liao and Grishman, 2010) leverage document-level information to enhance the performance of event extraction in a pipelined fashion, others propose a more structured framework for joint training of both trigger labeling and argument extraction (Li et al., 2013).

Other approaches explore neural networks on top of linguistic features employing architectures like Dynamic Multi-Pooling CNNs (DM-CNN) (Chen et al., 2015) and bidirectional RNNs (JRNN) with manually crafted features (Nguyen et al., 2016). A joint approach was proposed by Liu et al. (2018b) to extract multiple events based on syntactic graph convolution network. More recently, Zhang and Ji (2018) propose an approach based on inverse reinforcement learning using Generative Adversarial Networks (GAN) to alleviate mistakes related to ambiguous labels making the model less vulnerable to biased, supervised datasets like ACE2005.

However, the majority of the described approaches involves to some degree the use of linguistic features. This is labor intensive and requires rich external resources, which are not necessarily available for low-resource languages.

5.2 Cross-lingual Event Extraction

Previous works for cross-lingual event extraction conducted in a semi-supervised way range from purely supervised approaches to those using ma-

	MUSE	BERT
Davies is **leaving** to become chairman of the London School of Economics	Movement: Transport	Personnel: End-Position
The EU is set to release 20 million euros (US 21) million in immediate humanitarian aid ...	Justice: Release-Parole	Transaction: Transfer-Money
Palestinian **uprising** as Isreal removed all major checkpoints in the coastal territory.	Conflict: Demonstrate	Conflict: Attack
	BERT(mono)	**BERT(multi_all)**
لم يسلم ارسنال من الغرامة حيث فرضت عليه اللجنة ... "Arsenal has not been **released** from the fine ..."	O	Justice:Fine
ينبغي فورا ان تتحول الثورة الى نضال "The stone **revolution** must immediately turn into a fight."	O	Conflict:Attack
由于月之海已经宣布年底前要解散，所以使得... "Since 'the sea of the moon' has been announced to be **disbanded** before the end of the year, ... "	B-Business: Declare-Bankruptcy	Business: End-Org

Table 3: Examples of trigger extraction mislabeled by MUSE but correctly labeled by BERT and those missed/mislabeled with monolingual training only and corrected with multilingual BERT model.

chine translation techniques or word alignment data. Feng et al. (2016) propose a language-independent approach that doesn't require any linguistic feature engineering. However, this approach still requires equally abundant labeled data for different languages and implies the need to train a new model for each language independently.

Hsi et al. (2016) exploit both language-dependent and language-independent features in the form of universal features such as universal dependencies, limited bilingual dictionaries and aligned multilingual word embeddings to train a model with multiple languages. However, this work lags behind in terms of the neural approach used and doesn't investigate the effectiveness of leveraging multiple source languages.

Liu et al. (2018b) propose gated cross-lingual attention as a mechanism to exploit the inherent complementarity of multilingual data which helps with data scarcity and trigger disambiguation. However, this approach relies on machine translation which suffers from error propagation.

5.3 Cross-lingual Tasks

Cross-lingual embeddings are of practical usefulness in many tasks in natural language processing (NLP) and information extraction (IE). In each case, a model is trained on one language and transferred to unseen languages. Downstream applications on which they are applied include part-of-speech (POS) tagging (Cohen et al., 2011), cross-lingual document classification ((Klementiev et al., 2012); (Schwenk and Li, 2018)) named entity recognition (Xie et al., 2018). More recently, BERT was developed as an extension to the transformer architecture and achieved significant improvement in performance for many NLP tasks.

The gain in performance associated with multilingual training is what encouraged us to explore this methodology on event trigger extraction. To the best of our knowledge, there is no prior work adopting conventional or contextualized multilingual embeddings for event trigger detection.

6 Conclusion

In this work, we propose a cross-lingual approach for event trigger extraction using a direct transfer of annotation framework based on multilingual embeddings. Compared to previous approaches, our approach doesn't rely on hand-crafted linguistic features or machine translation.

We evaluate this approach using event trigger extraction architectures with type-based unsupervised embeddings (FastText and MUSE) and supervised embeddings tuned to the context (BERT). Our results for both English and Chinese show competitive performance with baselines on the ACE2005 benchmark even in the zero-shot learning scheme. Although results using MUSE are lower for English, they are on par with Chinese baselines and better for Arabic compared to BERT.

We observe a generous boost in performance when English is added to the target language, and when all languages are combined together to train one cross-lingual model, especially for Arabic. Our result are promising compared to both feature-based approaches and cross-lingual approaches based on machine translation.

Acknowledgment

This material is based on research sponsored by DARPA under agreement number FA8750-18- 2-0014. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Chen Chen and Vincent Ng. 2012. Joint modeling for Chinese event extraction with rich linguistic features. In *Proceedings of COLING 2012*, pages 529–544, Mumbai, India. The COLING 2012 Organizing Committee.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176. Association for Computational Linguistics.

Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 50–61, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, MarcÁurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *CoRR*, abs/1710.04087.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 66–71, Berlin, Germany. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1127–1136. Association for Computational Linguistics.

Andrew Hsi, Yiming Yang, Jaime Carbonell, and Ruochen Xu. 2016. Leveraging multilingual training for limited resource event extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1201–1210, Osaka, Japan. The COLING 2016 Organizing Committee.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262. Association for Computational Linguistics.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012: Technical Papers*, pages 1459–1474, Mumbai, India.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797, Uppsala, Sweden. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2018a. Event detection via gated multilingual attention mechanism. In *AAAI*, pages 4865–4872. AAAI Press.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2993–2999. AAAI Press.

Xiao Liu, Zhunchen Luo, and Heyan Huang. 2018b. Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309. Association for Computational Linguistics.

Holger Schwenk and Xian Li. 2018. A corpus for multilingual document classification in eight languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge RNN and tensor-based argument interaction. In *AAAI*, pages 5916–5923. AAAI Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Christopher Walker. 2006. Ace 2005 multilingual training corpus ldc2006t06. In *Linguistic Data Consortium*, Philadelphia, United States of America.

Yonghui Wu, Mike Schuster, and Zhifeng Chen et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379, Brussels, Belgium. Association for Computational Linguistics.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–5294, Florence, Italy. Association for Computational Linguistics.

Tongtao Zhang and Heng Ji. 2018. Event extraction with generative adversarial imitation learning. *CoRR*, abs/1804.07881.

Deep Structured Neural Network for Event Temporal Relation Extraction

Rujun Han*, **I-Hung Hsu***, **Mu Yang**
{rujunhan, ihunghsu, yangmu} @isi.edu
Aram Galstyan, Ralph Weischedel, Nanyun Peng
{galstyan, weisched, npeng} @isi.edu
Information Sciences Institute, University of Southern California

Abstract

We propose a novel deep structured learning framework for event temporal relation extraction. The model consists of 1) a recurrent neural network (RNN) to learn scoring functions for pair-wise relations, and 2) a structured support vector machine (SSVM) to make joint predictions. The neural network automatically learns representations that account for long-term contexts to provide robust features for the structured model, while the SSVM incorporates domain knowledge such as transitive closure of temporal relations as constraints to make better globally consistent decisions. By jointly training the two components, our model combines the benefits of both data-driven learning and knowledge exploitation. Experimental results on three high-quality event temporal relation datasets (TCR, MATRES, and TB-Dense) demonstrate that incorporated with pre-trained contextualized embeddings, the proposed model achieves significantly better performances than the state-of-the-art methods on all three datasets. We also provide thorough ablation studies to investigate our model.

1 Introduction

Event temporal relation extraction aims at building a graph where nodes correspond to events within a given text, and edges reflect temporal relations between the events. Figure 1a illustrates an example of such graph for the text shown above. Different types of edges specify different temporal relations: the event **filed** is *SIMULTANEOUS* with **claiming**, **overruled** is *BEFORE* **claiming**, and **overruled** is also *BEFORE* **filed**. Temporal relation extraction is beneficial for many downstream tasks such as question answering, information retrieval, and natural language generation. An event graph can potentially be leveraged to help time-series forecasting and provide guidances for natural language generation. The CaTeRs dataset (Mostafazadeh et al., 2016) which annotates temporal and causal relations is constructed for this purpose.

A major challenge in temporal relation extraction stems from its nature of being a *structured*

*Both U.S. and British officials **filed** objections to the court's jurisdiction in 1995, **claiming** Security Council resolutions **imposed** on Lybia to force the suspects' extradition **overruled** a 1971 Convention which gives Libya the right to try the men.*

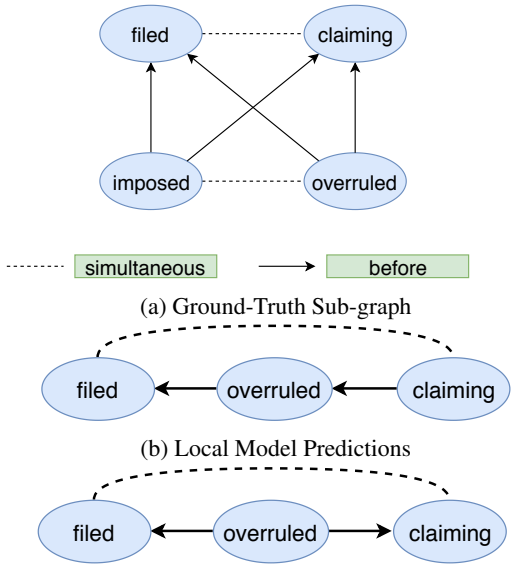


Figure 1: An illustration of a paragraph with its partial temporal graph. (a) shows the Ground-Truth temporal graph, in which case temporal relation *SIMULTANEOUS* and *BEFORE* are presented. (b) and (c) are the local and structured predictions, respectively, for three of the event pairs in (a). Local predictions are incompatible with temporal transitivity rule: **overruled** has to be *BEFORE* **claiming** if **overruled** is *BEFORE* **filed** and **filed** is *SIMULTANEOUS* with **claiming**. The structured model achieves coherence by reversing the temporal order between **claiming** and **overruled**.

* The authors contribute equally, alphabetical order.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 666–676
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

prediction problem. Although a relation graph can be decomposed into individual relations on each event pair, any *local* model that is not informed by the whole event graph will usually fail to make globally consistent predictions, thus degrading the overall performance. Figure 1b gives an example where the *local model* classifies the relation between **overruled** and **claiming** incorrectly as it only considers pairwise predictions: graph temporal transitivity constraint is violated given the relation between **filed** and **claiming** is *SIMULTANEOUS*. In Figure 1c, the *structured model* changes the prediction of relation between **overruled** and **claiming** from *AFTER* to *BEFORE* to ensure compatibility of all predicted edge types.

Prior works on event temporal relation extraction mostly formulate it as a pairwise classification problem (Bethard, 2013; Laokulrat et al., 2013; Chambers, 2013; Chambers et al., 2014) disregarding the global structures. Bramsen et al. (2006a); Chambers and Jurafsky (2008); Do et al. (2012); Ning et al. (2018b,a) explore leveraging global *inference* to ensure consistency for all pairwise predictions. There are a few prior works that directly model global structure in the training process (Yoshikawa et al., 2009; Ning et al., 2017; Leeuwenberg and Moens, 2017). However, these structured models rely on hand-crafted features using linguistic rules and local-context, which cannot adequately capture potential long-term dependencies between events. In the example shown in Figure 1, **filed** occurs in much earlier context than **overruled**. Thus, incorporating long-term contextual information can be critical for correctly predicting temporal relations.

In this paper, we propose a novel deep structured learning model to address the shortcomings of the previous methods. Specifically, we adapt the structured support vector machine (SSVM) (Finley and Joachims, 2008) to incorporate linguistic constraints and domain knowledge for making joint predictions on events temporal relations. Furthermore, we augment this framework with recurrent neural networks (RNNs) to learn long-term contexts. Despite the recent success of employing neural network models for event temporal relation extraction (Tourille et al., 2017a; Cheng and Miyao, 2017; Meng et al., 2017; Meng and Rumshisky, 2018), these systems make pairwise predictions, and do not take advantage of problem structures.

We develop a joint end-to-end *training* scheme that enables the feedback from global structure to directly guide neural networks to learn representations, and hence allows our deep structured model to combine the benefits of both data-driven learning and knowledge exploitation. In the ablation study, we further demonstrate the importance of each global constraints, the influence of linguistic features, as well as the usage of contextualized word representations in the local model.

To summarize, our main contributions are:

- We propose a deep SSVM model for event temporal relation extraction.
- We show strong empirical results and establish new state-of-the-art for three event relation benchmark datasets.
- Extensive ablation studies and thorough error analysis are conducted to understand the capacity and limitations of the proposed model, which provide insights for future research on temporal relation extraction.

2 Related Work

Temporal Relation Data. Temporal relation corpora such as TimeBank (Pustejovsky et al., 2003) and RED (O'Gorman et al., 2016) facilitate the research in temporal relation extraction. The common issue in these corpora is missing annotation. Collecting densely annotated temporal relation corpora with all event pairs fully annotated has been reported to be a challenging task as annotators could easily overlook some pairs (Cassidy et al., 2014; Bethard et al., 2007; Chambers et al., 2014). TB-Dense dataset mitigates this issue by forcing annotators to examine all pairs of events within the same or neighboring sentences. Recent data construction efforts such as MATRES (Ning et al., 2018a) and TCR (Ning et al., 2018b) further enhance the data quality by using a multi-axis annotation scheme and adopting start-point of events to improve inter-annotator agreements. However, densely annotated datasets are relatively small both in terms of number of documents and event pairs, which restricts the complexity of machine learning models used in previous research.

Event Temporal Relation Extraction The series of TempEval competitions (Verhagen et al., 2007, 2010; UzZaman et al., 2013) attract many research interests in predicting event temporal relations. Early attempts (Mani et al., 2006; Verha-

gen et al., 2007; Chambers et al., 2007; Verhagen and Pustejovsky, 2008) only use local pairwise classification with hand-engineered features. Later efforts, such as ClearTK (Bethard, 2013), UTTime (Laokulrat et al., 2013), and NavyTime (Chambers, 2013) improve earlier work by feature engineering with linguistic and syntactic rules. A noteworthy work, CAEVO (Chambers et al., 2014), builds a pipeline with ordered sieves. Each sieve is either a rule-based classifier or a machine learning model; sieves are sorted by precision, i.e. decisions from a lower precision classifier cannot contradict those from a higher precision model.

More recently, neural network-based methods have been employed for event temporal relation extraction (Tourille et al., 2017a; Cheng and Miyao, 2017; Meng et al., 2017; Han et al., 2019a) which achieved impressive results. However, they all treat the task as a pairwise classification problem. Meng and Rumshisky (2018) considered incorporating global context for pairwise relation predictions, but they do not explicitly model the *output* graph structure for event temporal relation.

There are a few prior works exploring structured learning for temporal relation extraction (Yoshikawa et al., 2009; Ning et al., 2017; Leeuwenberg and Moens, 2017). However, their local models use hand-engineered linguistic features. Despite the effectiveness of hand-crafted features in previous research, the design of features usually fails to capture long-term context in the discourse. Therefore, we propose to enhance the hand-crafted features with contextual representations learned through RNN models and develop an integrated joint training process.

3 Methods

We adapt the notations from Ning et al. (2018a), where $\mathcal{R}$ denotes the set of all possible relations; $\mathcal{E}$ denotes the set of all event entities.

3.1 Deep SSVM

Our deep SSVM model adapts the SSVM loss as

$$\mathcal{L} = \sum_{n=1}^{l} \frac{1}{M^n} \big[\max\big(0, \Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n) + S(\hat{\boldsymbol{y}}^n; \boldsymbol{x}^n) - S(\boldsymbol{y}^n; \boldsymbol{x}^n)\big)\big] + C||\Phi||^2, \quad (1)$$

where Φ denotes model parameters, n indexes instances, M^n is the number of event pairs in instance n. $\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n$ denote the gold and predicted global assignments for instance n, each of

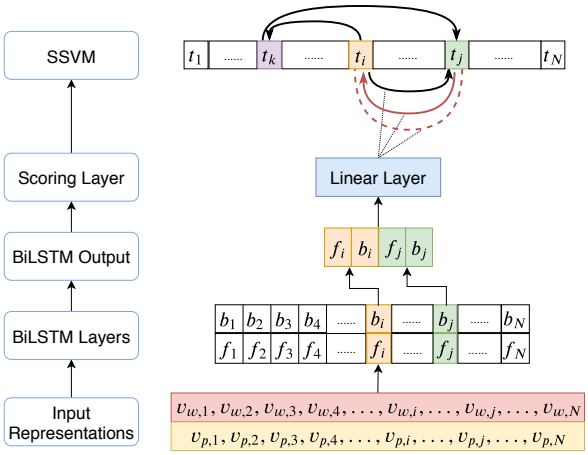


Figure 2: An overview of the proposed deep structured event relation extraction framework. The input representations consist of BERT representations ($v_{w,k}$) and POS tag embeddings ($v_{p,k}$). They are concatenated to pass through BiLSTM layers and classification layers to get pairwise local scores. Incompatible local pairwise prediction (denoted by red lines) is corrected by the SSVM layer. Edge notation follows Figure 1 and $t_1, ...t_N$ denote tokens in the input sentence.

which consists of M^n one hot vectors $\boldsymbol{y}^n_{i,j}, \hat{\boldsymbol{y}}^n_{i,j} \in \{0,1\}^{|\mathcal{R}|}$ representing true and predicted relation labels for event pair i, j respectively. $\Delta(\boldsymbol{y^n}, \hat{\boldsymbol{y}}^n)$ is a distance measurement between the gold and the predicted assignments; we simply use hamming distance. C is a hyper-parameter to balance the loss and the regularizer, and $S(\boldsymbol{y}^n; \boldsymbol{x}^n)$ is a pairwise scoring function to be learned.

The intuition behind the SSVM loss is that it requires the score of gold output structure $\boldsymbol{y}^n$ to be greater than the score of the best output structure under the current model $\hat{\boldsymbol{y}}^n$ with a margin $\Delta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n)$[1], or else there will be some loss.

The major difference between our deep SSVM and the traditional SSVM model is the scoring function. Traditional SSVM uses a linear function over hand-crafted features to compute the scores, whereas we propose to use a RNN for estimation.

3.2 RNN-Based Scoring Function

We introduce a RNN-based pair-wise scoring function to learn features in a data-driven way and capture long-term context in the input. The local neural architecture is inspired by prior work in entity relation extraction such as Tourille et al. (2017b). As shown in Figure 2, the input layer consists of word representations and part-of-

[1]Note that if the best prediction is the same as the gold structure, the margin is zero.

speech (POS) tag embeddings of each token in the input sentence, denoted as $v_{w,k}$ and $v_{p,k}$ respectively.[2] The word representations are obtained via pre-trained BERT (Devlin et al., 2018)[3] model and are fixed throughout training, while the POS tag embeddings are tuned. The word and POS tag embeddings are concatenated to represent an input token, and then fed into a Bi-LSTM layer to get contextualized representations.

We assume the events are labeled in the text and use indices i and j to denote the tokens associated with an event pair $(i, j) \in \mathcal{EE}$ in the input sentences of length N. For each event pair (i, j), we take the forward and backward hidden vectors corresponding to each event, namely f_i, b_i, f_j, b_j to encode the event tokens. These hidden vectors are then concatenated to form the input to the final linear layer to produce a softmax distribution over all possible pair-wise relations, which we refer to as RNN-based **scoring function**.

3.3 Inference

The inference is needed both during training to obtain $\hat{\boldsymbol{y}}^n$ in the loss function (Equation 1), as well as during the test time to get globally compatible assignments. The inference framework is established by constructing a global objective function using scores from local model and imposing several global constraints: symmetry and transitivity as in Bramsen et al. (2006b); Chambers and Jurafsky (2008); Denis and Muller (2011); Do et al. (2012); Ning et al. (2017); Han et al. (2019b), as well as linguistic rules and temporal-causal constraints proposed by Ning et al. (2018a) to ensure global consistency. In this work, we incorporate the **symmetry**, **transitivity**, and **temporal-causal** constraints.

Objective Function. The objective function of the global inference maximizes the score of global assignments as specified in Equation 2[4].

$$\hat{y} = \arg\max \sum_{(i,j)\in\mathcal{EE}} \sum_{r\in\mathcal{R}} y^r_{i,j} S(y^r_{i,j}; \boldsymbol{x}) \quad (2)$$

$$\text{s.t. } y^r_{i,j} \in \{0, 1\}\,, \sum_{r\in\mathcal{R}} y^r_{i,j} = 1,$$

where $y^r_{i,j}$ is a binary indicator specifying if the global prediction is equal to a certain label $r \in \mathcal{R}$ and $S(y^r_{i,j}, \boldsymbol{x}), \forall r \in \mathcal{R}$ is the scoring function obtained from the local model. The output of the global inference $\hat{y}$ is a collection of optimal label assignments for all event pairs in a fixed context. The constraint following immediately from the objective function is that the global inference should only assign one label to each pair of sample inputs.

Symmetry and Transitivity constraint. The symmetry and transitivity constraints are used across all models and experiments in the paper. They can be specified as follows:

$$\forall (i, j), (j, k) \in \mathcal{EE}, y^r_{i,j} = y^{\bar{r}}_{j,i}, \text{ (symmetry)}$$

$$y^{r_1}_{i,j} + y^{r_2}_{j,k} - \sum_{r_3 \in Trans(r_1, r_2)} y^{r_3}_{i,k} \leq 1. \text{ (transitivity)}$$

Intuitively, the symmetry constraint forces two pairs with opposite order to have reversed relations. For example, if $r_{i,j}$ = *BEFORE*, then $r_{j,i}$ = *AFTER*. Transitivity constraint rules that if (i, j), (j, k) and (i, k) pairs exist in the graph, the label (relation) prediction of (i, k) pair has to fall into the transitivity set specifying by (i, j) and (j, k) pairs. The full transitivity table can be found in Ning et al. (2018a).

Temporal-causal Constraint. The temporal-causal constraint is used for the TCR dataset which is the only dataset in our experiments that contains causal pairs and it can written as:

$$y^c_{i,j} = y^{\bar{c}}_{j,i} \leq y^b_{i,j}, \ \forall (i, j) \in \mathcal{EE},$$

where c and $\bar{c}$ correspond to the label *CAUSES* and *CAUSED_BY*, and b represents the label *BEFORE*. This constraint specifies that if event i causes event j, then i has to occur before j. Note that this constraint only has 91.9% accuracy in TCR data (Ning et al., 2018a), but it can help improve model performance based on our experiments.

3.4 Learning

We develop a two-state learning approach to optimize the neural SSVM. We first train the local scoring function without feedback from global constraints. In other words, the local neural network model is optimized using only pair-wise relations in the first stage by minimizing cross-entropy loss. During the second stage, we switch to the global objective function in Equation 1 and

[2] Following the convention of event relation prediction literature (Chambers et al., 2014; Ning et al., 2018a,b), we only consider event pairs that occur in the same or neighboring sentences, but the architecture can be easily adapted to the case where inputs are longer than two sentences.

[3] We use pre-trained bert-base-uncased model from https://github.com/huggingface/pytorch-transformers.

[4] The objective function is specified on the instance level.

	TB-Dense	MATRES	TCR
# of Documents			
Train	22	22	20
Dev	5	5	0
Test	9	9	5
# of Pairs			
Train	4032	1098	1992
Dev	629	229	0
Test	1427	310	1008

Table 1: Data Overview

	TB-Dense	MATRES	TCR
Local Model			
hid_size	60	40	30
dropout	0.5	0.7	0.5
BiLSTM layers	1	2	1
learning rate	0.002	0.002	0.002
Structured Learning			
learning rate	0.05	0.08	0.08
decay	0.7	0.7	0.9

Table 2: Best hyper-parameters

re-optimize the network to adjust for global properties[5]. We denote the local scoring model in the first stage as local model, and the final model as global model in the following sections.

4 Experimental Setup

In this section, we describe the three datasets that are used in the paper. Then we define the evaluation metrics. Finally, we provide details regarding our model implementation and experiments.

4.1 Data

Experiments are conducted on TB-Dense, MATRES and TCR datasets and an overview of data statistics are shown in Table 1. We focus on event relation, thus, all numbers refer to $\mathcal{EE}$ pairs[6]. Note that in all three datasets, event pairs are always annotated by their appearance order in text, i.e. given a labeled event pair (i, j), event i always appears prior to event j in the text. Following Meng et al. (2017), we augment event pairs with flipped-order pairs. That is, if a pair (i, j) exists, pair (j, i) is also added to our dataset with the opposite label. The augmentation is applied to training and development split, but test set remains unaugmented[7].

TB-Dense (Cassidy et al., 2014) is based on TimeBank Corpus but addresses the sparse-annotation issue in the original data by introducing the *VAGUE* label and requiring annotators to label all pairs of events/times in a given window.

MATRES (Ning et al., 2018b) is based on TB-Dense data, but filters out non-verbal events. The authors project events on multiple axes and only keep those in the main-axis. These two factors explain the large decrease of event pairs in Table 1. Start-point temporal scheme is adopted when outsourcing the annotation task, which contributes to the performance improvement of machine learning models built on this dataset .

TCR (Ning et al., 2018a) follows the same annotation scheme for temporal pairs in MATRES. It is also annotated with causal pairs. To get causal pairs, the authors select candidates based on EventCausality dataset (Do et al., 2011).

4.2 Evaluation Metrics

To be consistent with the evaluation metrics used in baseline models, we adopt two slightly different calculations of metrics.

Micro-average For all datasets, we compute the micro-average scores. For densely annotated data, the micro-average metric should share the same precision, recall and F1 scores. However, since VAGUE pairs are excluded in the micro-average calculations of TCR and MATRES for fair comparisons with the baseline models, the micro-average for precision, recall and F1 scores are different when reporting results for the two datasets.

Temporal Awareness (TE3) For TB-Dense dataset, TE3 evaluation scheme (UzZaman et al., 2013) is also adopted in previous research (Ning et al., 2017, 2018a). TE3 score not only takes into account of the number of correct pairs but also capture how "useful" a temporal graph is. We report this score for TB-Dense results only. For more details of this metric, please refer to the original paper (UzZaman et al., 2013).

4.3 Implementation Details

Since our work focuses on event relations, we build our models to predict relations between $\mathcal{EE}$ pairs only when conducting experiments. Thus, all micro-average F1 scores only consider $\mathcal{EE}$ pairs. Note that there are also time entities labeled in the TB-Dense denoted as $\mathcal{T}$. $\mathcal{ET}$ and $\mathcal{TT}$ pairs are

[5]We experiment with optimizing SSVM loss directly, but model performance degrades significantly. We leave further investigation for future work.

[6]For TCR, we also include causal pairs in the table.

[7]It is noted that if symmetric constraint is applied, scores for testing on augmented or unaugmented set are equal.

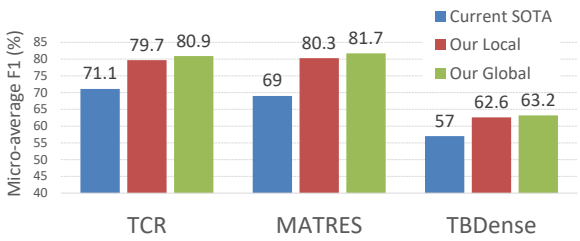


Figure 3: Model Performance (F1 Score) Overview. Our local and global models' performances are averaged over 3 different random seeds for robustness.

generally easier to predict using rule-based classifiers or date normalization technique (Do et al., 2012; Chambers et al., 2014; Mirza and Tonelli, 2016). To be consistent with the baseline models (Ning et al., 2018a,b) for TB-Dense data, we add $\mathcal{ET}$ and $\mathcal{TT}$ pairs for TE3 evaluation metric[8].

In the two-stage learning procedure, the local model is trained by minimizing cross-entropy loss with Adam optimizer. We use pre-trained BERT embedding with 768 dimensions as the input word representations and one-layer MLP as the classification layer. As for the structured learning stage, we observe performance boost by switching from Adam optimizer to SGD optimizer with decay and momentum[9]. To solve ILP in the inference process specified in Section 3.3, we leverage off-the-shelf solver provided by Gurobi optimizer, i.e. the best solutions from the Gurobi optimizer are inputs to the global training.

The hyper-parameters are chosen by the performance on the development set[10], and the best combination of hyper-parameters can be found in Table 2. We run experiments on 3 different random seeds and report the average results.

Note that for TCR data, we need a separate classifier for causal relations. Because of small amount of causal pairs, we simply build an independent final linear layer apart from the original linear layer in Figure 2. In other words, there are two final linear layers: only one of them is active when training temporal or causal pairs.

5 Results and Analysis

Figure 3 shows an overview of our model performance on three different datasets. As the chart illustrates, our RNN-based local models outperform state-of-the-art (SOTA) results and the global models further improve the performance over local models across all three datasets.

[8]We rely on annotated data to distinguish different pair types, i.e. $\mathcal{EE}$, $\mathcal{ET}$ and $\mathcal{TT}$ are assumed to be given.

[9]The weight decay in SGD is exactly the value C in Equation 1. We set the momentum in SGD as 0.9 in all datasets.

[10]We randomly select 4 documents from the training set as development set for TCR.

	Local Model			Global Model		
	P	**R**	**F1**	**P**	**R**	**F1**
Before	82.1	86.9	84.3	81.3	90.0	85.4
After	67.1	73.2	69.7	70.9	70.9	70.9
Simultaneous	0.0	0.0	0.0	0.0	0.0	0.0
Vague	0.0	0.0	0.0	0.0	0.0	0.0
Micro-average	77.1	82.5	79.7	78.2	83.9	**80.9****
Ning et al. (2018a)						71.1

Table 3: Model Performance Breakdown for TCR. To make fair comparison, we exclude VAGUE pairs in Micro-average score, which is why P, R and F1 are different. **indicates global model outperforms local model with p-value < 0.01 per McNemar's test.

	Local Model			Global Model		
	P	**R**	**F1**	**P**	**R**	**F1**
Before	79.7	88.1	83.6	80.1	89.6	84.6
After	70.5	83.3	76.3	72.3	84.8	78.0
Simultaneous	0.0	0.0	0.0	0.0	0.0	0.0
Vague	0.0	0.0	0.0	0.0	0.0	0.0
Micro-average	76.2	84.9	80.3	77.4	86.4	**81.7***
Ning et al. (2018b)						69

Table 4: Model Performance Breakdown for MATRES. Again, we exclude VAGUE pairs in Micro-average score. * indicates global model outperforms local model with p-value < 0.05 per McNemar's test.

5.1 TCR

Detailed model performances for the TCR dataset are shown in Table 3. We only report model performance on temporal pairs. Both of our local and global models outperform the baseline. Our global model is able to improve overall model performance by more than 1.2% over our local model; per McNemar's test, this improvement is statistically significant (with p-value < 0.01).

5.2 MATRES

Detailed model performances for the MATRES dataset performances can be found in Table 4. Similar to TCR, both our local and structured models outperform this baseline and the global model is able to improve overall model performance by 1.4%; per McNemar's test, this improvement is statistically significant (with p-value < 0.05).

5.3 TB-Dense

Table 5 shows the breakdown performance for all labels as well as the improvement from local to

	Local Model			Global Model		
	P	R	F1	P	R	F1
Before	73.5	52.7	61.3	71.1	58.9	64.4
After	71.6	60.8	65.3	75.0	55.6	63.5
Includes	17.5	4.8	7.4	24.6	4.2	6.9
Is_Include	69.1	4.4	8.0	57.9	5.7	10.2
Simultaneous	0.0	0.0	0.0	0.0	0.0	0.0
Vague	57.9	81.5	67.7	58.3	81.2	67.8
Micro-average			62.6			**63.2**
Chambers et al. (2014)						49.4
Cheng and Miyao (2017)						52.9
Meng and Rumshisky (2018)						**57.0**
TE3 Metrics						
$\mathcal{EE}$ only	62.1	61.9	62.2	62.7	58.9	**62.5**
$+\mathcal{ET}, \mathcal{TT}$	58.6	63.6	61.0	59.0	64.0	61.4
Ning et al. (2018a)						52.1

Table 5: Model Performance Breakdown for TB-Dense (all values are percentage). Upper Table: for event pairs only, we adopt standard Micro-average score. Lower Table: TE3 refers to the temporal awareness score adopted by TE-3 Workshop. To make fair comparison with Ning et al. (2018a), we add CEAVO predictions on $\mathcal{ET}$ and $\mathcal{TT}$ pairs back into the calculation.

global model by adopting the two-stage structured learning method in TB-Dense dataset. Both our local and global models are able to outperform previous SOTA in micro-average metric (reported by Meng and Rumshisky (2018)) or in TE3 metric (results from Ning et al. (2018a)).

Per McNemar's test, the improvements from local to global model only has p-value $= 0.126$, so we are not able to conclude that the improvement is statistically significant. We think one of the reasons is the large share of VAGUE pairs (42.6%). VAGUE pairs make our transitivity rules less conclusive. For example, if $\mathcal{R}(e1, e2)$ = BEFORE and $\mathcal{R}(e2, e3)$ = VAGUE, $\mathcal{R}(e1, e3)$ can be any relation types. Moreover, this impact is magnified by our local model's prediction bias towards VAGUE pairs. As we can see in Table 5, the recall score for VAGUE pairs are much higher than other relation types, whereas precision score is moderate. Our global model leverages local output structure to enforce global prediction consistency, but when local predictions contain many VAGUE pairs, it introduces lots of noise too.

To make fair comparison between our model and the best reported TE3 F1 score from Ning et al. (2018a), we follow their strategy and add CAEVO system's predictions on $\mathcal{TT}$ and $\mathcal{ET}$ pairs in the evaluation. The scores are shown in Table 5. Our overall system outperforms the baseline over 10% for both micro-average and TE3 F1 scores.

5.4 Error Analysis

To understand why both the local and structured models make mistakes, we randomly sample 50 pairs from 345 cases where both models' predictions are incorrect among all 3 random seeds. We analyze these pairs qualitatively and categorize them into four cases as shown in Table 6, with each case (except other) paired with an example.

The first case illustrates that correct prediction requires broader contextual knowledge. For example, the gold label for **transition** and **discuss** is *BEFORE*, where the nominal event **transition** refers to a specific era in history that ends before **discuss** in the second sentence. Human annotators can easily infer this relation based on their knowledge in history, but it is difficult for machines without prior knowledge. We observe this as a very common mistake especially for pairs with nominal events. As for the second case shows that negation can completely change the temporal order. The gold label for the event pair **planned** and **plans** is *AFTER* because the negation token **no** postpones the event **planned** indefinitely. Our models do not pick up this signal and hence predict the relation as *VAGUE*.

Finally, "intention" events could make temporal relation prediction difficult (Ning et al., 2018b). Case 3 demonstrates that our models could ignore the "intention" tokens such as **aimed at** in the example and hence make an incorrect prediction *VAGUE* between **doubling** and **signed**, whereas the true label is *AFTER* because **doubling** is an intention that has not occurred.

6 Ablation Studies

Although we have presented strong empirical results, the isolated contribution of each component of our model has not been investigated. In this section, we perform a though ablation study to understand the importance of structured constraints, linguistic features, and the BERT representations.

6.1 Effect of the structured constraints

One of our core claims is that our learning benefits from modeling the structural constraints of event temporal graph. To study the contribution of structured constraints, we provide an ablation study on two constraints that are applied to all three datasets: Symmetry and Transitivity.

A straightforward ablation study on symmetric constraint is to remove it from our global inference

Case 1 (32%): Connection with broader context
The program also calls for coordination of economic reforms and joint improvement of social programs in the two countries, where many people have become impoverished during the chaotic post - Soviet **transition** to capitalism. Kuchma also planned to visit Russian gas giant Gazprom, most likely to **discuss** Ukraine's DLRS 1.2 billion debt to the company.
Case 2 (20%): Negation
Annan has **no** trip **planned** so far. Meanwhile, Secretary of State Madeleine Albright, Berger and Defense Secretary William Cohen announced **plans** to travel to an unnamed city in the us heartland next week, to explain to the American people just why military force will be necessary if diplomacy fails.
Case 3 (14%): Intention Axis
A major goal of Kuchma's four - day state visit was the signing of a 10-year economic program **aimed at doubling** the two nations' trade turnover, which fell to DLRS 14 billion last year, down DLRS 2.5 billion from 1996. The two presidents on Friday **signed** the plan, which calls for cooperation in the metallurgy, fuel, energy, aircraft building, missile, space and chemical industries.
Case 4: (34%) Other

Table 6: Error Categories and Examples in TB-Dense

step. However, even though we eliminate symmetric constraint explicitly in global inference, it is utilized implicitly in our data augmentation steps (Section 4.1). To better understand the benefits of the symmetry constraints, we study both the contribution of *explicitly* applying symmetry constraint in our SSVM as well as its *implicit* impact in data augmentation.

Hence, in this section, we view a pair with original order and flipped order as different instances for learning and evaluation. We denote the pairs with original order as "**forward**" data, their flipped-order counterparts as "**backward**" data, and their combinations as "**both-way**" data.

We train four additional models to study the impacts of symmetry and transitivity constraints: 1) local model trained on forward data; 2) global model with transitivity constraint trained on forward data; 3) local model trained on both-way data; 4) global model with transitivity constraint trained on both-way data, denoted as M_1, M_2, M_3, M_4 respectively. M_1 and M_2 are models that do not apply any symmetric property; M_3 and M_4 are models that utilize symmetric property implicitly.

Additionally, evaluation setup should be re-scrutinized if we remove the symmetry constraints. In the standard evaluation setup of prior works, evaluation is only performed on the pairs with their original order (forward data) in text. This evaluation assumes a model will work equally well for both forward and backward

	TB-Dense		Matres		TCR	
	$\overrightarrow{Test}$	$\overleftrightarrow{Test}$	$\overrightarrow{Test}$	$\overleftrightarrow{Test}$	$\overrightarrow{Test}$	$\overleftrightarrow{Test}$
$M_1: \overrightarrow{L}$	62.9	61.9	80.4	74.7	80.5	75.7
$M_2: \overrightarrow{L}+T$	**63.2**	62.0	**81.7**	75.7	**81.0**	76.3
$M_3: \overleftrightarrow{L}$	62.6	62.7	80.3	80.4	79.7	79.6
$M_4: \overleftrightarrow{L}+T$	63.1	63.0	81.4	81.4	80.3	80.2
$\overleftrightarrow{L}+S+T$ (Proposed)	**63.2**	**63.2**	**81.7**	**81.7**	80.9	**80.9**

Table 7: Ablation over global constraints: Symmetry and Transitivity. Test is conducted on forward test set and both-way test set, which are denoted as $\overrightarrow{Test}$ and $\overleftrightarrow{Test}$ respectively. The local models trained on forward data and both-way data are denoted as $\overrightarrow{L}$ and $\overleftrightarrow{L}$. Symmetry and Transitivity constraints are denoted as S and T. The results demonstrate that symmetry and transitivity constraints both improve model's performance.

data, which certainly holds when we explicitly impose symmetry constraints. However, as we can observe in the later analysis, this assumption fails when we remove symmetry constraints. To demonstrate the improvement of model robustness over backward data, we propose to test the model on both forward and both-way data. If a model is robust, it should perform well on both scenarios.

We summarize our analysis of the results in Table 7 (F1 scores) as follows:

- Impact of Transitivity: By comparing M_1 with M_2 and M_3 with M_4, the consistent improvements across all three datasets demonstrate the effectiveness of global transitivity constraints.
- Impact of Implicit Symmetry (data augmentation): Examining the contrast between M_1 and M_3 as well as M_2 and M_4, we can see significant improvements in both-way evaluation despite slight performance drops in forward evaluation. These comparisons imply that data augmentation can help improve model robustness. Note that Meng and Rumshisky (2018) leveraged this data augmentation trick in their model.
- Impact of Explicit Symmetry: By comparing the proposed model with M_4, the consistent improvements across all datasets demonstrate the benefit of using explicit symmetry property.
- Model Robustness: Although M_1 and M_2 show competitive results when evaluated on forward test data, their performance degrade

	local- w/ feat.	global- w/ feat.	local- w/o feat.	global- w/o feat.
TB-Dense	62.5	63.0	62.6	**63.2**
MATRES	81.4	**81.7**	80.3	**81.7**
TCR	79.5	80.7	79.7	**80.9**

Table 8: Ablation study on linguistic feature usage. Additional linguistic features do not lead to significant improvement and even hurt performance in 2 out of 3 datasets. The results show that our proposed framework is semantic-rich and capable of avoiding the usage of additional linguistic feature.

significantly in the both-way evaluation. In contrast, the proposed model achieves strong performances in both test scenarios (best F1 scores except for one), and hence proves the robustness of our proposed method.

6.2 Effect of linguistic features

Previous research establish the success of leveraging linguistic features in event relation prediction. One advantage of leveraging contextualized word embedding is to provide rich semantic representation and could potentially avoid the usage of extra linguistic features. Here, we study the impact of incorporating linguistic features to our model by using simple features provided in the original datasets: token distance, tense and polarity of event entities. These features are concatenated with the Bi-LSTM hidden states before the linear layer (i.e. f_i, b_i, f_j, b_j in Figure 2). Table 8 shows the F1 scores of our local and global model using or not using linguistic features respectively. These additional features likely cause over-fitting and hence do not improve model performance across all three datasets we test. This set of experiments show that linguistic features do not improve the predicting power of our current framework.

6.3 Effect of BERT representations

In this section, we explore the impact of contextualized BERT representations under our deep SSVM framework. We replace BERT representations with the GloVe (Pennington et al., 2014) word embeddings. Table 9 shows the F1 scores of our local model and global model using BERT and GloVe[11] respectively. BERT improves the performance with a significant margin. Besides, even without BERT representations, our RNN-based local model and the deep structured global model still outperform (MATRES and TCR) or are comparable with (TB-Dense) current SOTA. These results confirm the improvements of our method.

[11]For GloVe model, additional linguistic features are used.

	previous SOTA	local- Glove	global- Glove	local- BERT	global- BERT
TB-Dense	57.0	56.6	57.0	62.6	**63.2**
MATRES	69.0	71.8	75.6	80.3	**81.7**
TCR	71.1	73.5	76.5	79.7	**80.9**

Table 9: Ablation over word representation: BERT vs GloVe. Although BERT representation largely contributes to the performance boost, our proposed framework remains strong and outperforms current SOTA approaches when GloVe is used.

7 Conclusion

In this paper, we propose a novel deep structured model based on SSVM that combines the benefits of structured models' ability to encode structure knowledge, and data-driven deep neural architectures' ability to learn long-range features. Our experimental results exhibit the effectiveness of this approach for event temporal relation extraction.

One interesting future direction is further leveraging commonsense knowledge, domain knowledge in temporal relation, and linguistics information to create more robust and comprehensive global constraints for structured learning. Another direction is to improve feature representations by designing novel neural architectures that better capture negation and hypothetical phrases as discussed in error analysis. We plan to leverage large amount of unannotated corpora to help event temporal relation extraction as well.

Acknowledgments

This work is partially funded by DARPA Contracts W911NF-15-1-0543 and an NIH R01 (LM012592). The authors thank the anonymous reviewers for their helpful suggestions and members from USC PLUS lab for early feedbacks. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

References

Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 10–14. Association for Computational Linguistics.

Steven Bethard, James H. Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *Proceedings of the International Conference on Semantic Computing*, ICSC '07, pages 11–18, Washington, DC, USA. IEEE Computer Society.

P. Bramsen, P. Deshpande, Y. K. Lee, and R. Barzilay. 2006a. Inducing temporal graphs. In *EMNLP*, Sydney, Australia.

Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006b. Inducing temporal graphs. In *EMNLP*, Sydney, Australia.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 501–506. Association for Computational Linguistics.

Nate Chambers. 2013. Navytime: Event and time ordering from raw text. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 73–77, Atlanta, Georgia, USA. Association for Computational Linguistics.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. In *ACL*.

Nathanael Chambers and Dan Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *EMNLP*, Honolulu, United States.

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 173–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional lstm over dependency paths. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 1–6.

Pascal Denis and Philippe Muller. 2011. Predicting globally-coherent temporal structures from texts via endpoint inference and graph decomposition. In *IJCAI*, Barcelone, Spain.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. Minimally supervised event causality identification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *EMNLP*, Jeju, Korea.

Thomas Finley and Thorsten Joachims. 2008. Training structural svms when exact inference is intractable. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 304–311, New York, NY, USA. ACM.

Rujun Han, Mengyue Liang, Bashar Alhafni, and Nanyun Peng. 2019a. Contextualized word embeddings enhanced event temporal relation extraction for story understanding. *arXiv preprint arXiv:1904.11942*.

Rujun Han, Qiang Ning, and Nanyun Peng. 2019b. Joint event and temporal relation extraction with shared representations and structured prediction.

Natsuda Laokulrat, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Uttime: Temporal relation classification using deep syntactic features. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 88–92, Atlanta, Georgia, USA. Association for Computational Linguistics.

Artuur Leeuwenberg and Marie-Francine Moens. 2017. Structured learning for temporal relation extraction from clinical records. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1150–1158.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 753–760, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yuanliang Meng and Anna Rumshisky. 2018. Context-aware neural model for temporal information extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Yuanliang Meng, Anna Rumshisky, and Alexey Romanov. 2017. Temporal information extraction for question answering using syntactic dependencies in an lstm-based architecture. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 887–896.

Paramita Mirza and Sara Tonelli. 2016. Catena: Causal and temporal relation extraction from natural language texts. In *ACL*, Osaka, Japan.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *NAACL*, San Diego, USA.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *EMNLP*, Copenhagen, Denmark.

Qiang Ning, Zhili Feng, Hao Wu, and Dan Roth. 2018a. Joint reasoning for temporal and causal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2278–2288. Association for Computational Linguistics.

Qiang Ning, Hao Wu, and Dan Roth. 2018b. A multi-axis annotation scheme for event temporal relations. In *ACL*. Association for Computational Linguistics.

Tim O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer event description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of 2nd Workshop on Computing News Storylines*, pages 47–56. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, and Lisa Ferro. 2003. The timebank corpus. In *Corpus linguistics*, pages 647–656.

Julien Tourille, Olivier Ferret, Aurelie Neveol, and Xavier Tannier. 2017a. Neural architecture for temporal relation extraction: a bi-lstm approach for detecting narrative containers. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 224–230.

Julien Tourille, Olivier Ferret, Xavier Tannier, and Aurélie Névéol. 2017b. Limsi-cot at semeval-2017 task 12: Neural architecture for temporal information extraction from clinical narratives. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 597–602. Association for Computational Linguistics.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 75–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the tarsqi toolkit. In *22Nd International Conference on on Computational Linguistics: Demonstration Papers*, COLING '08, pages 189–192, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 57–62, Stroudsburg, PA, USA. Association for Computational Linguistics.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 405–413. Association for Computational Linguistics.

Investigating Entity Knowledge in BERT with Simple Neural End-To-End Entity Linking

Samuel Broscheit
Data and Web Science Group, University of Mannheim, Germany
broscheit@informatik.uni-mannheim.de

Abstract

A typical architecture for end-to-end entity linking systems consists of three steps: mention detection, candidate generation and entity disambiguation. In this study we investigate the following questions: (a) Can all those steps be learned jointly with a model for contextualized text-representations, i.e. BERT (Devlin et al., 2019)? (b) How much entity knowledge is already contained in pretrained BERT? (c) Does additional entity knowledge improve BERT's performance in downstream tasks? To this end, we propose an extreme simplification of the entity linking setup that works surprisingly well: simply cast it as a per token classification over the entire entity vocabulary (over $700K$ classes in our case). We show on an entity linking benchmark that (i) this model improves the entity representations over plain BERT, (ii) that it outperforms entity linking architectures that optimize the tasks separately and (iii) that it only comes second to the current state-of-the-art that does mention detection and entity disambiguation jointly. Additionally, we investigate the usefulness of entity-aware token-representations in the text-understanding benchmark GLUE, as well as the question answering benchmarks SQUAD V2 and SWAG and also the EN-DE WMT14 machine translation benchmark. To our surprise, we find that most of those benchmarks do not benefit from additional entity knowledge, except for a task with very small training data, the RTE task in GLUE, which improves by 2%.

1 Introduction

The goal of entity linking is, given a knowledge base (KB) and unstructured data, e.g. text, to detect mentions of the KB's entities in the unstructured data and link them to the correct KB entry. The entity linking task is typically implemented by the following steps:

- Mention detection (MD): text spans of potential entity mentions are identified,
- Candidate generation (CG): entity candidates for each mention are retrieved from the KB,
- Entity disambiguation (ED): (typically) a mix of useful coreference and coherence features together with a classifier determine the entity link.

Durrett and Klein (2014) were the first to propose jointly modelling MD, CG and ED in a graphical model and could show that each of those steps are interdependent and benefit from a joint objective. Other approaches only model MD and ED jointly (Nguyen et al., 2016; Kolitsas et al., 2018), thus these architectures depend on a CG step after mention detection. Hachey et al. (2013); Guo et al. (2013); Durrett and Klein (2014) showed the influence of CG on entity linking, because it can be the coverage bottleneck, when the correct entity is not contained in the candidates for ED. Yamada et al. (2016, 2017) use a precomputed set of entity candidates published by Pershina et al. (2015) for their experiments on the CoNLL03/AIDA benchmark dataset (Hoffart et al., 2011), and due to this their experiments are comparable across studies with regards to the CG step. MD has a similar impact on entity linking performance, as it determines the upper bound of linkable mentions.

BERT (Devlin et al., 2019) is a deep self-attention-based architecture which is pretrained on large amounts of data with a language modelling objective. This model provides very rich linguistic text-representations that have been shown to be very useful for many NLP tasks. Since its appearance, BERT is being analyzed and applied in various domains (Beltagy et al., 2019; Lee et al., 2019). A recent study found that BERT automatically learns the NLP pipeline (Tenney et al., 2019),

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 677–685
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

i.e. a stack of increasingly higher level linguistic functions. Zhang et al. (2019) investigated injecting entity knowledge from noisy[1] automatic entity linking into the pretraining of BERT and they could show that this improves relation extraction.

In this study we investigate the following questions:

(a) Can BERT's architecture learn all entity linking steps jointly? We propose an extreme simplification of entity linking and cast it as a per token classification over the entire entity vocabulary, thus solving MD, CG and ED simultaneously (see Fig. 1). The entity vocabulary is based on the $700K$ top most frequent entities in English Wikipedia and the training data was derived from English Wikipedia texts. We first trained BERT-base-uncased on English Wikipedia (dubbed BERT+Entity) and then fine-tuned and evaluated it on an entity linking benchmark. We found that this worked surprisingly well for entity linking, even if we do not have any supervision on mention-spans, i.e. BIO tags. An error analysis with validation data revealed that only 3% of errors are purely due to span errors, while most errors are due to wrong *Nil* predictions which often coincided with entities being infrequent.

(b) How much entity knowledge is already contained in pretrained BERT? To investigate this question, we froze BERT and only trained the entity classifier of BERT+Entity on Wikipedia (dubbed Frozen-BERT+Entity), i.e. the resulting entity classifier is adjusted for entity mentions for which plain BERT already does assign distinct token representations, such that correct entity classification is possible. Then we fine-tuned and evaluated Frozen-BERT+Entity on an entity linking benchmark. We find that the performance of Frozen-BERT+Entity is 6% below BERT+Entity, showing that BERT+Entity has learned additional entity knowledge.

(c) Does additional entity knowledge improve BERT's performance in downstream tasks? Due to training BERT+Entity with a per token classification, the model is forced to assign distinct entity specific features to each token of an entity mention. Downstream tasks could exploit this, if additional entity information is necessary for them. We evaluated BERT+Entity in the natural language understanding benchmark GLUE (Wang et al., 2018), the question answering (QA) benchmarks SQUAD V2 (Rajpurkar et al., 2018) and SWAG (Zellers et al., 2018), and the machine translation benchmark EN-DE WMT14. We confirm the finding from Zhang et al. (2019) that additional entity knowledge is not beneficial for the GLUE benchmark. To our surprise, we also find that additional entity knowledge is neither helpful for the two QA datasets nor for machine translation. The only exception is the RTE task in GLUE in which BERT+Entity improves 2%. This dataset has just 0.5-2% of the training data of the two larger natural language inference datasets in GLUE.

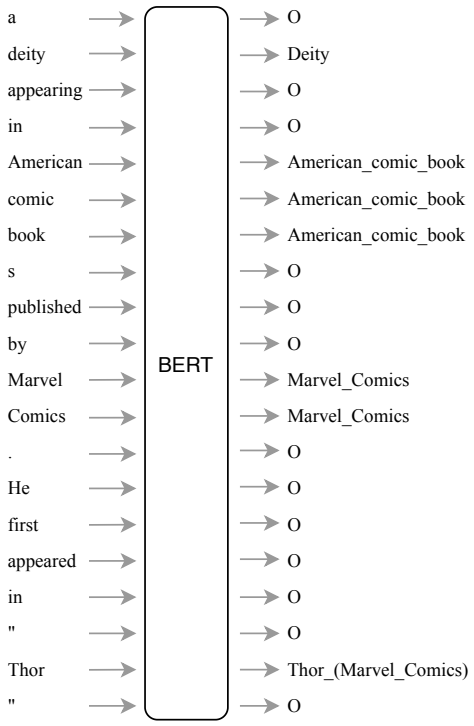


Figure 1: Illustrating the simple neural end-to-end entity linking setup. BERT+Entity predicts entity links per token, where "O" denotes a *Nil* prediction. The example shows how context can help to link *"Thor"* to *Thor_(Marvel_Comics)*.

Our contributions are: We are the first to study the latter questions. We are also the first to propose a fully neural model, that does MD, CG and ED all in one model, i.e. performing entity linking without any pipeline or any heuristics. We are also the first to propose to model entity linking as a token classification and show that this seems to be a viable option. We also uncover that there is a lack of tasks that evaluate additional entity knowledge in pretrained language models.

[1]TagMe's performance on various benchmark datasets ranges from 37% to 72%. F1 (Kolitsas et al., 2018)

2 Related Work

Entity Linking Durrett and Klein (2014) is the work that is closest to our approach, although not neural. In their approach they model interactions between the MD, CG and ED tasks jointly. They find that the joint objective is beneficial, such that each task improves. They also note that there is no natural order of the tasks and they should interact freely. Their approach to CG is to learn to generate queries to the KB. Nguyen et al. (2016) also propose jointly modelling MD and ED with a graphical model and show that it improves ED performance and is more robust. Kolitsas et al. (2018) recently published their study in which they propose the first neural model to learn MD and ED jointly. Their proposed method is to overgenerate mentions and prune them with a mention-entity dictionary. The ED step reasons over the remaining mentions if and to what they link to. However, modern approaches for solving natural language tasks operate on neural text-representations, and the approaches discussed so far only yield entity-links. Yamada et al. (2016, 2017) was the first to investigate neural text representations and entity linking, but their approach is limited to ED.

Pretrained Language Models ULMFIT (Howard and Ruder, 2018), ELMO (Peters et al., 2018), BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) are modern language models that are very deep and wide (for NLP) and are pretrained on large amounts of data. They provide very rich text representations that have shown to improve many NLP tasks by just replacing the static word embeddings with deep contextualized word embeddings. As Peters et al. (2019) show, further training the deep language models alongside the model that uses the embeddings as input can be helpful, for which the term "finetuning" is used. The current trend in research is to investigate all aspects of these language models, seeking insights in their inner workings (Tenney et al., 2019), or their application to various domains (Beltagy et al., 2019; Lee et al., 2019). In this study, we investigate the factual information in form of entities that is contained in BERT, seeking to understand to what degree this information is already identifiable in BERT and if the entity knowledge can be improved.

3 End-To-End Neural Entity-Linking

In this section we describe the BERT+Entity, which is a is straightforward extension of BERT, however, as with the original BERT, the main challenge lies in designing the training scheme, i.e. in our case the creation of the training data. Our goal for the experiments is to evaluate, if we can learn candidate generation, thus a desiderata is to make the entity vocabulary as large as possible to be comparable to other studies. The text data and the entity linking annotations are derived from Wikipedia by exploiting intra-Wikipedia links. This yields the challenge that the annotations for entity links from Wikipedia are assumed to be incomplete, i.e. not every entity mention in Wikipedia is linked, which we hypothesize can be detrimental during training.

3.1 Model

Our model is based on BERT, which is a deep self-attention-based architecture (Vaswani et al., 2017) that was trained on large amounts of text. Its training objective is two-fold: (a.) predict missing tokens from sentences, and (b.) classify if a second sentence was an adjacent sentence. The input and output token vocabulary are sub-words, i.e. the vocabulary is computed from the training data by determining the $30K$ most frequent character sequences, excluding spaces. Devlin et al. (2019) made several pretrained BERT models publicly available. They differ in size — i.e. token embedding size and self-attention layer depth — and whether the token vocabulary is cased or uncased. BERT+Entity is a straightforward extension on top of BERT, i.e. we initialize BERT with the publicly available weights from the BERT-base-uncased model and add an output classification layer on top of the architecture. Given a contextualized token, the classifier computes the probability of an entity link for each entry in the entity vocabulary. Formally, let d be BERT's token embedding size, and $E \in \mathbb{R}^{|KB| \times d}$ the entity classification layer, with $|KB|$ being the number of entities in the KB, V is the sub-word vocabulary, $c_i = BERT(h)[i]$ is the i-th contextualized token computed by BERT from context $h = [v_1, v_2, ..., v_{i-1}, v_i, v_{i+1}, ..., v_m]$ with each $v \in V$. Consequently, the probability $p(j|v, h)$ of word v — which is the i-th token in context h — linking to entity j is computed by $\sigma(E_j c_i)$, where σ is the sigmoid function.

3.2 Training Data

The entity vocabulary and training data are derived from English Wikipedia texts[2]. We used an extended version of WikiExtractor[3] to extract the text spans that are associated with an internal Wikipedia link to use as annotation, e.g. in the sentence *"The first Thor was all about introducing Asgard"*, the text span *"Thor"* links to *https://en.wikipedia.org/wiki/Thor_(film)*. BERT is originally trained with sentences. However, for entity linking, a larger context can help to disambiguate entity mentions, which is why we select text fragments of such a length, that they span multiple sentences. For later use we collect (m, e) tuples of entities e and their mention m. This yields a set M of potentially linkable strings and also lets us compute the conditional probability $p(e|m)$ based on the $\#(m, e)$ counts.

Handling incomplete annotation A challenge in using the Wikipedia links as annotation is that most entities do not have all their mentions annotated, i.e. often only the first appearance in an article is linked. We hypothesize that learning a classifier on such skewed data would yield a skewed model. Our approach to counter missing annotations is two-fold: (i) We only select text fragments that contain a minimum count of annotated Wikipedia links. (ii) To account for unlinked mentions in the fragments we use a Trie-based matcher[4] to annotate all occurrences of linkable strings that we collected in M. As entity links we annotate all possible entities this mention could link to but only with the conditional probability $p(e|m)$, with the goal that the model remembers a context independent entity prior. One issue is that due to the incomplete annotation, the $\#(e, m)$ counts yield $p(Nil|\text{"}\textit{United States}\text{"}) > 0$, i.e. the mention *"United States"* has a large non-zero probability to link to nothing. Based on the assumption that the mentions of the most popular entities should always link to something, we compute the average of the probability of linking to *Nil* for the $k = 1000$ most frequent entities

$$\bar{p}_{Nil} = \frac{1}{k} \sum_{j} \frac{\#(m_j, Nil)}{\#m_i}.$$

and use $\#(m_i, Nil) - \frac{\bar{p}_{Nil}}{(1-\bar{p}_{Nil})} * \#(m_i, e_*)$ to discount $\#(m_i, Nil)$ such that $p(Nil|\text{"}\textit{United States}\text{"}) \approx 0$, i.e. the model should always link "United States" and mentions of less frequent entities get an increase in probability to link to something.

4 Entity Linking Experiments

In the experiments we want to investigate how the simple neural end-to-end entity linking model BERT+Entity performs, i.e. if it learns something additional on-top of BERT. Additionally, we investigated if the entity-aware token-representations are useful for downstream tasks. We also discuss the main engineering challenges training with such a large entity vocabulary.

4.1 Data

Wikipedia We report two settings which differ in size of the entity vocabulary, size of the fragments and minimum number of entities per fragments. The first setting was the initial study, and the second one is a follow up study in which we changed settings that potentially could improve entity linking performance.

Setting I: We keep the $700K$ top most frequent entities from the $\approx$ $6M$ entities in Wikipedia, i.e. we chose the entity vocabulary as large as it was technically feasible with regards to memory and training speed. To put it into context, the CoNLL03/AIDA entity linking benchmark contains $23,5K$ entities in 1300 documents. We are missing 30 entities from CoNLL03/AIDA that only appear less than 10 times in the Wikipedia training data. We chunk the Wikipedia texts into fragments with a length of 110 tokens and an overlap of 20 tokens with the previous and following fragment. We only keep fragments that contain at least 1 infrequent linked entity or at least 3 frequent ones. This yields $8,8M$ training instances from which we take 1000 each for validation and testing.

Setting II: We keep the $500K$ top most frequent entities, which is comparable to the entity vocabulary of Kolitsas et al. (2018) and we have to add ≈ 1000 entities from CoNLL03/AIDA to the entity vocabulary to be able to evaluate our model on that benchmark. We increase the fragment size to 250 tokens and keep fragments that contain at least 1 linked entity but keep at most 500 fragments per entity. This yields $2,4M$ training instances from which we take 500 each for validation and testing.

[2]From a enwiki Wikipedia dump from 20.06.2017.

[3]https://github.com/samuelbroscheit/wikiextractor-wikimentions

[4]https://github.com/vi3k6i5/flashtext

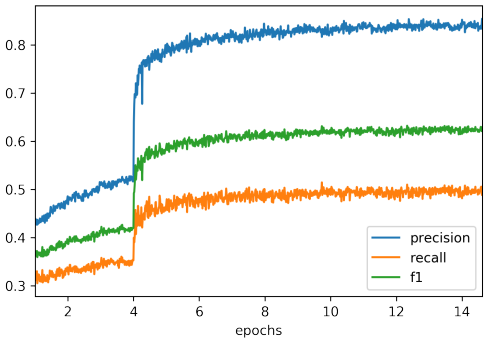


Figure 2: Per token classification InKB scores on the validation data during training on the Wikipedia dataset in Setting II for 40 days. The jump at the 4-th epoch happens when we switch from training Frozen-BERT+Entity to BERT+Entity, i.e. when we start fine-tuning BERT.

Entity Linking Benchmark To evaluate on a commonly used benchmark dataset we use CoNLL03/AIDA. It is the biggest manually annotated ED dataset. It contains 946 documents in training, 216 in validation (testa/AIDA-VALID) and 231 in test (testb/AIDA-TEST).

4.2 Training

We use a multi-class classification over the entity vocabulary, i.e. the label y vector for one token v_i is defined by

$$y_{ij} = p(j|v_i), \text{ for } j \in \{1, .., ||KB||\}.$$

However, computing the loss over the whole entity vocabulary would be infeasible, because the entity mention vocabulary is very large and the gradients for the entity classifier would exceed our GPU memory. Thus, to improve memory efficiency and increase convergence speed, we use negative sampling. After sampling text fragments for a batch b, we collected the set $N+_b$ of all true entities — according to the annotations discussed in Sec. 3.2 — that occurred in those text fragments. Ideally we would update the representations of those entities that do not occur in the set $N+_b$ which the model is erroneously the most confident about. To achieve this, we first performed a prediction for the text fragments in the current batch and collected for each token the top k predicted entities. We aggregated the entities' logits over the whole batch and sorted the entities by their aggregated logits into the list N_b- and removed from it any entity contained in N_b+. We join $N_b = N_b+ \cup N_b-$ and truncate N_b- such that $|N_b|$ equals a given maximum size. Each label vector y_i for token c_i from fragment C in batch b was now defined over the entities in N_b. Thus, we only predict over the corresponding subset of the entity embedding table, i.e. $\hat{E} = E(N_b)$. The loss for one fragment C in batch b was computed by

$$L = \frac{1}{|N_b| * |C|} \sum_{i}^{|C|} \sum_{j}^{|N_b|} -[y_{ij} \cdot \log \sigma(\hat{E}_j c_i) + (1 - y_{ij}) \cdot \log(1 - \sigma(\hat{E}_j c_i))].$$

For training on Wikipedia we used Adam (Kingma and Ba, 2015) with mini batch size 10, gradient accumulation over 4 batches, maximum label size 10240, the learning rate for BERT was $5e$-5 and for the entity classifier 0.01. In Setting I we train the model for 4 epochs, one epoch took five days with two TitanXp/1080Ti. In the first 1.5 epochs we train Frozen-BERT+Entity and then BERT+Entity. In Setting II we train the model for 14 epochs and one epoch took three days. In the first 3 epochs we train Frozen-BERT+Entity and then BERT+Entity.

For training on CoNLL03/AIDA we used Adam (Kingma and Ba, 2015) with mini batch size 10, gradient accumulation over 4 batches, maximum label size 1024, learning rates for BERT $5e$-5, dropout in BERT 0.2, and we freeze the token embeddings, the first two layers of BERT and the entity classifier. We train the remaining parameters for up to 30 epochs and perform early stopping according to strong match (see next Section). One epoch took seven minutes with one TITAN Xp/1080 Ti.

4.3 Performance Metrics

We compute the Micro InKB Precision, Recall and F1 metrics and we only consider entities as true, if they are in our KB. We compute a strong match, i.e. every token in the gold annotated span has to be classified correctly. We also report a weak match, which we define as at least one token in the gold annotated span having to link to the correct entity. This setting accounts for annotation inconsistencies, e.g. when the model and the annotation do not agree on which mention *"U.S. army"* or *"U.S."* to annotate (can be either way). We also report strong ED Precision@1, i.e. we ignore *Nil* predictions of the model and only evaluate the top ranked entity only for spans that have a gold entity.

		AIDA/testa			AIDA/testb		
		strong F1	weak F1	ED	strong F1	weak F1	ED
Kolitsas et al. (2018) indep. baseline		75.7	76.0	-	73.3	73.9	-
Kolitsas et al. (2018)		86.6	87.2	92.4	82.6	83.2	89.1
BERT		63.3	66.6	67.6	49.6	52.4	52.8
Setting I	Frozen-BERT+Entity	76.8	79.6	80.6	64.7	68.0	68.6
	BERT+Entity	82.8	84.4	86.6	74.8	76.5	78.8
Setting II	Frozen-BERT+Entity	76.5	80.1	79.6	67.8	71.9	67.8
	BERT+Entity	86.0	87.3	92.3	79.3	81.1	87.9

Table 1: Comparing entity linking results on CoNLL03/AIDA. *strong F1* and *weak F1* denote InKB F1 scores. *ED* is Precision@1 for InKB. Kolitsas et al. (2018) also study a neural model, however, they only model MD and ED. The independent baseline shows how their model performs when they use mentions detected by Stanford NLP. In Frozen-BERT+Entity BERT is not trained and only the entity classifier on-top is trained.

4.3.1 Results

In Table 1 we compare our results to the most recent results by Kolitsas et al. (2018) who studied a neural approach that does joint modelling of MD and ED, but not CG. They also provide a baseline in which they show how their classifier performs when MD and ED are independent, i.e. linking mentions detected by Stanford NLP.

For the reported results denoted only with BERT, the entity classifier is trained from scratch on CoNLL03/AIDA and BERT is finetuned. This shows the lower bound on this dataset, i.e. the amount of information that we can learn with BERT only from the CoNLL03/AIDA training data. Note, that this cannot generalize to entities that are not contained in training. The difference between BERT and Frozen-BERT+Entity shows the amount of entity knowledge that plain BERT already had, which it transferred in the entity classifier during training on Wikipedia. Finally, BERT+Entity is the proposed model, in which both BERT and the entity classifier have been trained on Wikipedia.

4.3.2 Discussion

Comparing BERT+Entity and Frozen-BERT+Entity we see that there is a significant amount of entity knowledge that BERT+Entity learns additionally to Frozen-BERT+Entity, i.e. training BERT+Entity increases the scores between 6%-10% depending on the score and dataset. However, it should also be noted that Frozen-BERT+Entity already shows an increase of 13%-16% over BERT, thus it already learns for many entities distinct features that enable the entity classifier to identify them. The improvement of Frozen-BERT+Entity in contrast to BERT on CoNLL03/AIDA shows that this pretraining generalizes to validation and test data. We can also observe that Setting II improves by a large margin over Setting I and comes very close to the results of Kolitsas et al. (2018). We conjecture that the biggest impact on the performance from changing the training from Setting I to Setting II, was due to the downsampling of the training data in favor of less frequent entities. This reduction of training data in Setting II — caused by capping the maximum amount of examples per entity — enabled us to run more epochs in less time, which might have improved the representations of less frequent entities.

Reason for error	#
no prediction	57
different than gold annotation	
no obvious reason	13
semantic close	4
lexical overlap	5
nested entity	5
gold annotation wrong	12
span error	3
unclear	1
	100

Table 2: Investigating the types of strong precision errors of BERT+Entity trained in Setting I on CoNLL03/AIDA (testa) on 100 randomly sampled strong precision errors from the validation dataset.

Task	Metric	BERT-BERT-Ensemble	BERT+Entity-Ensemble
CoLA	Matthew's corr.	59.92	59.97
SST-2	accuracy	92.73	92.43
MRPC	F1/accuracy	89.16	90.13
STS-B	Pearson/Spearman corr.	89.90	89.60
QQP	accuracy	91.64	91.21
MNLI	matched acc./mismatched acc.	84.96	84.78
QNLI	accuracy	91.21	91.15
RTE	accuracy	71.48	73.64
WNLI	accuracy	56.33	56.33
SQUAD V2	matched/mismatched	76.89/73.83	76.36/73.46
SWAG	accurracy	80.70	80.76
WMT14 EN-DE	BLEU	22.51	22.20

Table 3: Experiments on downstream tasks with BERT+Entity trained in Setting I. The first group are the GLUE tasks, then followed by SQUAD V2 and SWAG (for which only the dev set results are reported), and the results for machine translation WMT14 EN-DE.

When we compare BERT+Entity with the two results from Kolitsas et al. (2018), we observe that BERT+Entity improves over the baseline that models MD, CG and ED independently, and that BERT+Entity comes second to the current state-of-the-art in end-to-end entity linking. What can also be observed is that the performance of all models drops from AIDA/testa to AIDA/testb. For BERT+Entity, however, the drop is more severe, obviously the model overfits to some patterns in the training data that are present in the validation data, but not in the test data. We hypothesize that this might be due to some sport specific documents that make roughly $1/4$ of the dataset's mentions. However, without spoiling the test-set we cannot know for sure.

In Table 2 we performed an error analysis for the experiments for Setting I to learn what kind of strong precision errors are responsible for the performance of BERT+Entity. The largest source of errors was that BERT+Entity did predict *Nil* instead of an entity. We hypothesized that most of the *no prediction* errors are because those entities have only a low frequency in the training data, i.e. this could be solved by increasing the model size and improving the training time. Another source of error we observed was that the context size was too small due to the fragment size. A surprisingly positive result from the error analysis was that in only 3% a wrong span caused the error. Motivated by the observations we devised the follow-up experiment Setting II (see Section 4.1) in which we changed some of the settings to potentially solve the observed issues.

5 Downstream Tasks Experiments

In this section we discuss the downstream task results. We performed evaluations on the natural language understand task GLUE, the question answering tasks SQUAD V2 and SWAG and the machine translation benchmark EN-DE WMT14. We found that only in one of the subtasks of GLUE —the natural language inference tasks RTE— BERT+Entity performs better than BERT, for all other we can observe no such effect. The reported results are for Setting I, however, we repeated the experiments with Setting II and observed the same outcomes.

5.1 Model

For the tasks GLUE, SQUAD V2 and SWAG we extend hugginface's implementation[5] and concatenate the outputs of BERT and BERT+Entity (dubbed BERT+Entity-Ensemble) or two BERTs (dubbed BERT-BERT-Ensemble). For EN-DE WMT14 we use BERT (dubbed BERT-2Seq) or BERT+Entity (dubbed BERT+Entity-2Seq) as encoder and use a Transformer decoder by adapting fairseqs Pytorch Seq2Seq Transformer implementation (Ott et al., 2019).

[5]https://github.com/huggingface/pytorch-pretrained-BERT

5.2 Training

For the GLUE benchmark, SQUAD and SWAG we train the BERT+Entity-Ensemble and BERT-BERT-Ensemble for 3 epochs and use the default hyper-parameters from the implementation. The models BERT-2Seq and BERT+Entity-2Seq we train for 4 epochs, with Adam as optimizer and learning rate $5e$-5, max 1000 tokens per batch, clip gradient norm 0.1, dropout 0.2, label smoothing 0.1, and we keep the encoders BERT and BERT+Entity fixed for the first epoch and then train it together with the decoder.

5.3 Results

We find that the additional entity knowledge is not helpful in the evaluated tasks. The results in Table 3 show that, except for RTE, there seems to be no advantage in having additional entity knowledge. The question is, if this is (a) due to the entity overlap in training and testing such that also an entity unaware model can learn the necessary model, or (b) the entities are too scarce in the training data to make a difference, or (c) the tasks themselves do not require entity knowledge, i.e. other textual cues are enough. We leave those questions for future research.

6 Conclusion

In this study we investigated an extremely simplified approach to entity linking that worked surprisingly well and allowed us to investigate entity knowledge in BERT. Even when there is a gap to the current state-of-the-art in entity linking, we hypothesize that this gap can be closed with larger hardware capacity to scale up the model size and effective training time. Apart from that, the model is the first that performs entity linking without any pipeline or any heuristics, compared to all prior approaches. We found that with our approach we can learn additional entity knowledge in BERT that helps in entity linking. However, we also found that almost none of the downstream tasks really required entity knowledge, which is an interesting observation and an open question for future research.

Acknowledgments

The author would like to gratefully thank the NVIDIA corporation for the donation of a TITAN Xp GPU that was used in this research.

References

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *TACL*, 2:477–490.

Yuhang Guo, Bing Qin, Yuqin Li, Ting Liu, and Sheng Li. 2013. Improving candidate generation for entity linking. In *Natural Language Processing and Information Systems - 18th International Conference on Applications of Natural Language to Information Systems, NLDB 2013, Salford, UK, June 19-21, 2013. Proceedings*, pages 225–236.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating entity linking with wikipedia. *Artif. Intell.*, 194:130–150.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 782–792.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529, Brussels, Belgium. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So,

and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746.

Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2016. J-NERD: joint named entity recognition and disambiguation with rich linguistic features. *TACL*, 4:215–229.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 238–243.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019.*, pages 7–14.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4593–4601.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 353–355.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 250–259.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning distributed representations of texts and entities from knowledge base. *TACL*, 5:397–411.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 93–104.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. *to appear*.

Unsupervised Adversarial Domain Adaptation for Implicit Discourse Relation Classification

Hsin-Ping Huang
Department of Computer Science
The University of Texas at Austin
hsinping@cs.utexas.edu

Junyi Jessy Li
Department of Linguistics
The University of Texas at Austin
jessy@austin.utexas.edu

Abstract

Implicit discourse relations are not only more challenging to classify, but also to annotate, than their explicit counterparts. We tackle situations where training data for implicit relations are lacking, and exploit domain adaptation from explicit relations (Ji et al., 2015). We present an unsupervised adversarial domain adaptive network equipped with a reconstruction component. Our system outperforms prior works and other adversarial benchmarks for unsupervised domain adaptation. Additionally, we extend our system to take advantage of labeled data if some are available.

1 Introduction

Discourse relations capture the relationship between units of text—e.g., sentences and clauses—and are an important aspect of text coherence. While some relations are expressed explicitly with a discourse connective (e.g., "for example", "however"), relations are equally often expressed implicitly without an explicit connective (Prasad et al., 2008); in these cases, the relation needs to be inferred.

Resources for implicit discourse relations are scarce compared to the explicit ones, since they are harder to annotate (Miltsakaki et al., 2004). For example, among corpora annotated with discourse relations such as Arabic (Al-Saif and Markert, 2010), Czech (Poláková et al., 2013), Chinese (Zhou and Xue, 2015), English (Prasad et al., 2008), Hindi (Oza et al., 2009), and Turkish (Zeyrek et al., 2013), only the Chinese, English and Hindi corpora include implicit discourse relations (Prasad et al., 2014). In this low-resource scenario, Ji et al. (2015) proposed training with explicit relations via unsupervised domain adaptation, viewing explicit relations as a source domain with labeled training data, and implicit relations as a target domain with no labeled data. The domain gap between explicit and implicit relations is acknowledged by prior observations that the two types of discourse relations are linguistically dissimilar (Sporleder and Lascarides, 2008; Rutherford and Xue, 2015).

We present a new system for the unsupervised domain adaptation setup on the Penn Discourse Treebank (Prasad et al., 2008). Our system is based on Adversarial Discriminative Domain Adaptation (Tzeng et al., 2017), which decouples source domain training and representation mapping between source and target. We improve this framework by proposing a reconstruction component to preserve the discriminability of target features, and incorporating techniques for stabler training on textual data.

Experimental results show that even with a simple architecture for representation learning, our unsupervised domain adaptation system outperforms prior work by 1.4-2.3 macro F1, with substantial improvements on Temporal and Contingency relations. It is also superior to DANN (Ganin et al., 2016), an adversarial framework widely used in NLP (Chen et al., 2018; Gui et al., 2017; Zhang et al., 2017; Fu et al., 2017; Joty et al., 2017; Xu and Yang, 2017), by 5.7 macro F1.

Finally, we extend the system to incorporate in-domain supervision as it is sometimes feasible resource-wise to build a seed corpus that may not be large enough to train a fully supervised system. We simulate this scenario by enabling the system to jointly optimize over a varying number of labeled examples of implicit relations. Our system consistently outperforms two strong baselines.

2 Related Work

Sporleder and Lascarides (2008) and Rutherford

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 686–695
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

and Xue (2015) observed that explicit and implicit relations are linguistically dissimilar, warranting an unsupervised domain adaptation approach in Ji et al. (2015). They used a marginalized denoising autoencoder to obtain generalized feature representations across the source and target domains with a linear SVM as the classification model. Our system improves upon this work using an adversarial network; we further generalize our network to semi-supervised settings.

To supplement the training data of implicit discourse relations, prior works have used weak supervision from sentences with discourse connectives (Marcu and Echihabi, 2002; Sporleder and Lascarides, 2008; Braud and Denis, 2014; Ji et al., 2015), by analyzing connectives (Zhou et al., 2010a,b; Biran and McKeown, 2013; Rutherford and Xue, 2015; Braud and Denis, 2016; Wu et al., 2017), using a multi-task framework with other corpora (Lan et al., 2013; Liu et al., 2016; Lan et al., 2017), or utilizing cross-lingual data (Wu et al., 2016; Shi et al., 2017). The important distinction between this work and the research above is that these are supervised systems that used all of the annotated *implicit* annotation from PDTB during training, while exploring non-PDTB corpora for additional, noisy discourse cues; on the contrary, our main goal is to assume no labeled training data for implicit discourse relations.

Unsupervised domain adaptation with adversarial networks has become popular in recent years; this type of approach generates a representation for the target domain with the goal that the discriminator unable to distinguish between the source and target domains. Prior works proposed both generative approaches (Liu and Tuzel, 2016; Bousmalis et al., 2017; Sankaranarayanan et al., 2018; Russo et al., 2018) and discriminative approaches (Ganin et al., 2016; Tzeng et al., 2015, 2017). The discriminative DANN algorithm from Ganin et al. (2016) is frequently used in NLP tasks (Chen et al., 2018; Gui et al., 2017; Zhang et al., 2017; Fu et al., 2017; Joty et al., 2017; Xu and Yang, 2017). Our method builds upon Adversarial Discriminative Domain Adaptation (Tzeng et al., 2017), shown to outperform DANN in visual domain adaptation but has not been used in NLP tasks. The key differences between the two are discussed in Section 3.

Qin et al. (2017) adopted adversarial strategies to supervised implicit discourse classification. They train an adversarial model using implicit discourse relations with and without expert-inserted connectives. Note again that theirs is a fully supervised system using signals in addition to the implicit relation annotations themselves, while our main focus is unsupervised domain adaptation that does not train on implicit relations.

3 Model Architecture

To classify discourse relations, our system takes a pair of sentence arguments x as input, and outputs the discourse relation y between these two arguments. With unsupervised domain adaptation, we have examples (X_s, Y_s) from the source domain, i.e., explicit discourse relations, and unlabeled examples (X_t) from a target domain, i.e., implicit discourse relations.

We use ADDA (Tzeng et al., 2017) as our underlying framework for domain adaptation. ADDA first learns a discriminative representation for the classification task in the source domain, then learns a representation for the target domain that mimics the distribution of the source domain. The key insight here is asymmetric mapping, where the target representation is "updated" until it matches with the source, a process more similar to the original Generative Adversaial Networks (Goodfellow et al., 2014) than joint training as in DANN (Ganin et al., 2016). Intuitively, since ADDA learns distinct feature encoders for the source and target domains instead of using a shared encoder, the same network doesn't have to handle instances from different domains.

Summarized in Figure 1, we first pre-train a source encoder M_s and source classifier C (Section 3.1), then train the target encoder M_t (initialized with M_s) and discriminator D in an adversarial way, to minimize the domain discrepancy distance between the target representation distribution $M_t(X_t)$ and that of the source $M_s(X_s)$ (Section 3.2). Eventually, the target feature space is trained to match the source, and the source classifier C can be directly used on the target domain.

3.1 Base encoder and classifier

The source and target **encoders** M_s and M_t follow the same architecture; M_t is initialized to be M_s during adaptation. The encoders encode relation arguments into latent representations, and then feeds the representations into a **classifier** C to predict the discourse relation.

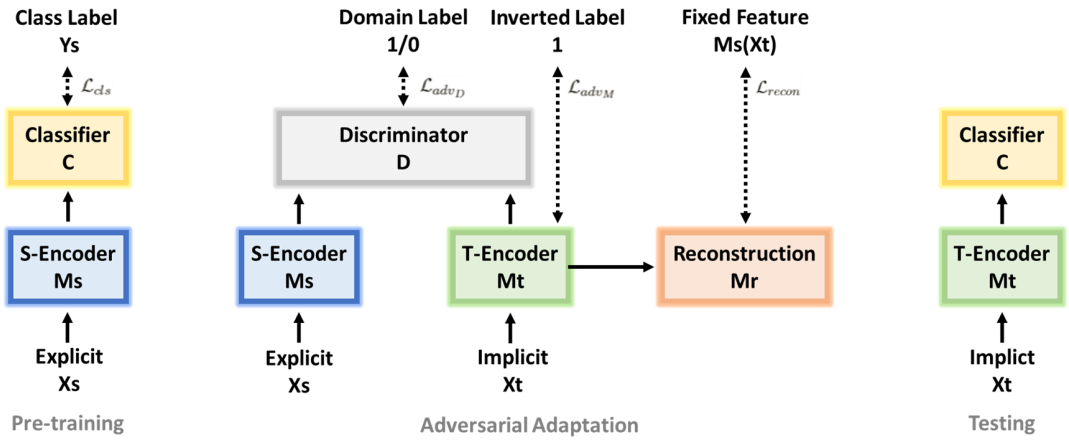


Figure 1: The framework of our proposed adversarial domain adaptation model, containing the pre-training stage, the adversarial adaptation stage, and the testing stage. The dashed box shows the supervised component.

Encoder The encoder generates a representation for each argument with an inner-attention Bidirectional LSTM (Yang et al., 2016) shared between the two arguments. Then, the representations of the two arguments are concatenated to form the final representation, shown in Figure 2.

Specifically, we encode each word in an argument into its word embeddings, which are fed into a BiLSTM, to get the hidden representations z_i using a fully-connected layer W_c on top of the concatenated hidden states $h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$. We then apply an attention mechanism to induce a distribution of weights over all tokens in the argument; the final argument representation Arg is a weighted sum of z_i based on the attention weights α_i:

$$
\begin{aligned}
z_i &= W_c h_i + b_c \\
u_i &= tanh(W_w h_i + b_w) \\
\alpha_i &= \frac{exp(u_i^T u_w)}{\sum_i exp(u_i^T u_w)} \\
Arg &= \sum_i \alpha_i z_i
\end{aligned} \tag{1}
$$

Where W_c, b_c, W_w, b_w, u_w are model parameters.

Classifier The classifier consists of a single fully-connected layer on top of the encoder, finished with a softmax classification layer.

The source encoder M_s and the classifier C are trained using a supervised loss:

$$
\begin{aligned}
&\min_{M_s, C} \mathcal{L}_{cls}(X_s, Y_s) = \\
&\mathbb{E}_{(x_s, y_s)} - \sum_k \mathbb{1}[k = y_s] \log C(M_s(x_s))
\end{aligned} \tag{2}
$$

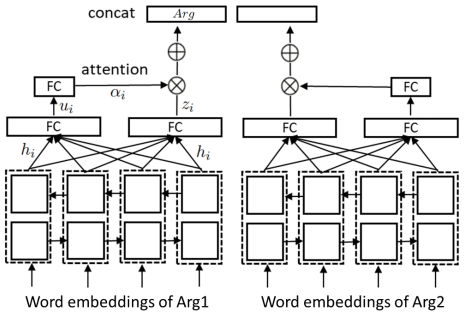


Figure 2: Neural structure of inner-attention BiLSTM to encode relation argument pairs.

3.2 Unsupervised adversarial domain adaptation

We then learn a target encoder M_t to generate features for the target data which can be classified with classifier C, without assuming labels Y_t in the target domain. This is achieved by training a domain discriminator D, which classifies whether a feature is from the source or the target domain, and the target encoder M_t, that produces features similar to the source domain features and tries to fool the discriminator to predict the incorrect domain label.

The discriminator D is optimized according to a standard supervised loss:

$$
\begin{aligned}
&\min_D \mathcal{L}_{adv_D}(X_s, X_t, M_s, M_t) = \\
&\quad - \mathbb{E}_{x_s}[\log D(M_s(x_s))] \\
&\quad - \mathbb{E}_{x_t}[\log (1 - D(M_t(x_t)))]
\end{aligned} \tag{3}
$$

D consists of two fully-connected layers on top of the encoder, finished with a softmax classification layer.

The target encoder M_t is optimized according to a standard GAN loss with inverted labels:

$$\min_{M_t} \mathcal{L}_{adv_M}(X_s, X_t, D) = -\mathbb{E}_{x_t}[\log D(M_t(x_t))] \quad (4)$$

Spectral normalization To stabilize the training of the discriminator, we employ spectral normalization, a weight normalization technique (Miyato et al., 2018), which controls the Lipschitz constant of the discriminator function by constraining the spectral norm of each layer. Spectral normalization is easy to implement without tuning any hyper-parameters and has a small additional computational cost.

Label smoothing We utilize label smoothing (Szegedy et al., 2016) to regularize the classifier during pre-training, which prevents the largest logit from becoming much larger than all others, and therefore prevents overfitting and makes the classifier, trained in the source domain, more adaptable.

For a source domain training example x_s with ground-truth label y_s and ground-truth distribution $q(k|x_s)$, the classifier computes the classification probability over relation classes as $p(k|x_s)$ for $k \in \{1...K\}$. With label smoothing, we replace the ground-truth label distribution $q(k|x_s)$ in the standard cross-entropy loss as a linear combination of $q(k|x_s)$ and a uniform distribution over classes $u(k) = 1/K$.

$$\min_{M_s,C} \mathcal{L}_{cls}(X_s, Y_s) = -\sum_k q'(k) \log\left(p(k)\right)$$
$$q'(k|x_s) = (1-\epsilon)\mathbb{1}[k = y_s] + \epsilon/K \quad (5)$$

Reconstruction loss In order to classify the target representations using the source classifier, the target encoder is trained to produce representations that mimic the source domain representations in the adversarial training stage. Since there is no supervised loss applied in this stage, the target encoder may lose its ability to produce discriminative features that are helpful during classification. We propose a reconstruction loss to preserve the discriminability of the target encoder when adversarially adapting its features.

Since we initialize the target encoder with the source encoder, the initial representation (before domain adaptation) of a target instance x_t is the representation of *target* instances produced by the *source* encoder $M_s(x_t)$ (which is then fixed). After training, $M_t(x_t)$ adapts to the source domain and becomes dissimilar to $M_s(x_t)$. The reconstruction loss encourages the target encoder to produce features that can be reconstructed back to $M_s(x_t)$ (Figure 3).

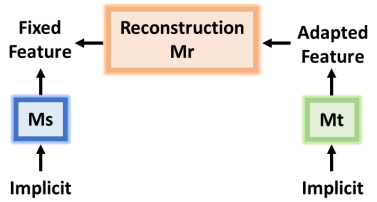


Figure 3: The reconstruction loss component augmenting our unsupervised adversarial domain adaptor.

For a target example x_t, we learn a reconstruction mapping M_r that maps the target representation $M_t(x_t)$ to $M_s(x_t)$:

$$x_t \rightarrow M_t(x_t) \rightarrow M_r(M_t(x_t)) \approx M_s(x_t) \quad (6)$$

The target encoder M_t and the reconstruction mapping M_r are optimized jointly with a reconstruction loss:

$$\min_{M_t,M_r} \mathcal{L}_{recon}(X_t, M_s) = \mathbb{E}_{x_t}[||M_r(M_t(x_t)) - M_s(x_t)||_2^2] \quad (7)$$

M_r consists of three fully-connected layers on top of the encoder.

Unsupervised objective For unsupervised domain adaptation, our full objective is:

$$\begin{aligned}\mathcal{L}^{unsup}(X_s, Y_s, X_t, M_s, M_t, D) = & \\ \min_{M_s,C} \mathcal{L}_{cls}(X_s, Y_s) + & \\ \min_{D} \mathcal{L}_{adv_D}(X_s, X_t, M_s, M_t) + & \\ \min_{M_t} \mathcal{L}_{adv_M}(X_s, X_t, D) + & \\ \min_{M_t,M_r} \mathcal{L}_{recon}(X_t, M_s) & \quad (8)\end{aligned}$$

3.3 Training

Summarized in Algorithm 1, the training procedure consists of three stages: pre-training, adversarial adaptation, and testing. During pre-training, we train the source encoder M_s and the classifier C according to Eq.(2). In the adversarial adaptation stage, we alternately train the discriminator D, target encoder M_t, and reconstruction mapping

Algorithm 1: Adversarial Adaptation

Input: explicit sentences with labels $\{x_s, y_s\}$
implicit sentences without labels $\{x_t\}$
Notations: source encoder M_s, classifier C,
target encoder M_t,
reconstruction mapping M_r,
domain discriminator D

1 Train M_s, C through Eq.(2) with $\{x_s, y_s\}$;
2 Initialize M_t as M_s;
3 **Repeat**
4 Train D through Eq.(3) with $\{x_s, x_t\}$ and train M_t through Eq.(4) with $\{x_t\}$;
5 Train M_t, M_r through Eq.(7) with $\{x_t\}$;
Output: M_t and C for relation prediction

M_r according to Eq.(3), Eq.(4), Eq.(7). Finally, we test the model using the target encoder M_t and classifier C. The steps (lines 4 and 5) in the Repeat loop execute once in one iteration, and we optimized the model in two-step units.

4 Unsupervised Domain Adaptation Experiments

We first evaluate our model for the default task: unsupervised domain adaptation from explicit discourse relations to implicit discourse relations.

4.1 Settings

Data We train and test our model on the PDTB, following the experimental setup of Ji et al. (2015). The test examples are implicit relation instances from PDTB sections 21-22. The explicit training set consists of explicit examples from sections 02-20 and 23-24, and the explicit development set consists of the explicit examples from sections 00-01. The implicit training set and the implicit development set consist of examples from the same sections as the explicit sets. Evaluation is done for the first-level relations—Temporal, Comparison, Contingency, and Expansion. Table 1 summarizes the statistics of the four top-level implicit and explicit discourse relations in the PDTB.

Training Details We early-stopped training for both stages before total convergence if the macro F1 on the development set does not improve. During pre-training (Line 1 in Algorithm 1), we train and validate the model on the explicit training and development set. Early stopping happened after around 20 epochs. During adversarial adaptation (lines 4 and 5 in Algorithm 1), we train the model on the explicit and implicit training sets *without relation labels* Y_t, and validate on the implicit development set[1]. Early stopping happened after around 5 epochs (with lines 4 and 5 executed once in each epoch).

	Explicit		Implicit		
Relation	Train	Dev	Train	Dev	Test
Temporal	2904	288	704	68	54
Contingency	2792	181	3622	276	287
Comparison	4674	366	2104	146	191
Expansion	5342	450	7394	556	651

Table 1: The number of examples of the four top level discourse relations in PDTB 2.0.

Model configuration The hyperparameters, as well as the number of fully connected layers for the classifier C, discriminator D and the reconstruction mapping M_r, are all set according to the performance on the development sets. We first set the hyper-parameters of the encoders M_s, M_t and classifier C based on development performance during the pre-training stage. Then, we set the hyper-parameters of D and M_r based on development performance of the adaptation stage.

We use GloVe (Pennington et al., 2014) for word embeddings with dimension 300. The maximum argument length is set to 80. The encoder contains an inner-attention BiLSTM with dimension 50, producing a representation with dimension 200 for each example. The discriminator D consists of 2 hidden layers with 200 and 200 neurons on each layer. The reconstruction mapping M_r contains 3 hidden layers with 120, 15 and 120 neurons on each layer. The label smoothing parameter ϵ is 0.1. We use Adam (Kingma and Ba, 2015) with learning rate 1e-4 for the base encoder and classifier, and 1e-6 for the adversarial domain adapter. We use SGD optimizer with learning rate 1e-2 for the reconstruction component. All the models were implemented using PyTorch (Paszke et al., 2017) and adapted from Conneau et al. (2017).

4.2 Systems

We experiment with three settings:

Implicit → Implicit A supervised implicit discourse relation classifier using the base encoder and classifier, optimizing the standard cross-entropy loss, using the full implicit training and

[1]Using a development set in the target domain is common in unsupervised domain adaptation (Ganin et al., 2016; Liu and Tuzel, 2016; Tzeng et al., 2017; Bousmalis et al., 2017; Russo et al., 2018)

	Temporal	Contingency	Comparison	Expansion	Macro F_1
Implicit → Implicit	25.53	41.02	30.35	65.38	40.57
Explicit → Implicit	22.22	22.35	23.06	57.86	31.37
+Domain Adaptation	30.62	42.71	25.00	52.23	37.64
+Spectral Normalization	30.20	45.42	21.90	58.72	39.06
+Label Smoothing	**31.58**	46.40	24.64	58.03	40.16
+Reconstruction	31.25	**48.04**	25.15	59.15	**40.90**
(Ji et al., 2015)	19.26	41.39	25.74	68.08	38.62
+weak supervision	20.35	42.25	**26.32**	**68.92**	39.46
DANN	26.19	34.20	25.74	54.70	35.21

Table 2: Per-class and macro average F1 (%) of unsupervised domain adaptation from explicit to implicit relations.

development sets. This model does not use the explicit relations.

Explicit → Implicit A discourse relation classifier using the explicit training set and implicit development set, optimizing the standard cross-entropy loss. This serves as a baseline without any domain adaptation.

Domain adaptation Our full adaptation model trained on the explicit training set, and adapted to the implicit training set without relation labels. We perform ablation study with different extensions describe in Section 3: spectral normalization, label smoothing, and reconstruction loss.

For *benchmarking*, we train an unsupervised domain adaptation system using **DANN** (Ganin et al., 2016), which jointly learns domain-invariant representations and the classifier and is often used in NLP (c.f. Section 2). We use the same encoder, classifier and discriminator structures, with parameters tuned on the implicit development data. The system is optimized using Adam with learning rate 2e-4 and the adaptation parameter 0.25, chosen between 0.01 and 1 on a logarithmic scale.

4.3 Results

To evaluate our model, we train four-way classifiers and report **per-class and macro F1 scores**. Table 2 tabulates the experimental results for unsupervised domain adaptation.

We also show reported results from Ji et al. (2015). Even though they trained four binary classifiers (instead of doing multi-class classification), it is the only prior work exploring unsupervised domain adaptation for implicit discourse relation classification. We include two settings: their best system with labeled data from PDTB explicit relations only (and an implicit development set), and their system with additional weak supervision from non-PDTB sources.[2]

Our full system achieves the best average F1 measure, a 9.53% absolute increase from Explicit → Implicit. It also performs 2.28% better than Ji et al. (2015)'s model trained without weak supervision, and 1.44% better than their model trained with weak supervision. The full system achieved an average F1 comparable to the supervised Implicit → Implicit, while Ji et al. (2015)'s models did not. Comparing with DANN, our system achieved superior performance for 3 of the 4 relations, showing that training target representations and the classifier in two stages outperforms doing both jointly.

The largest improvements from the Explicit → Implicit baseline are from Temporal (from 22.22 to 31.25) and Contingency (from 22.35 to 48.04) relations. Our system performs ∼11% better for Temporal, and ∼6% better for Contingency, than Ji et al. (2015)'s binary classifiers. The Comparison and Expansion relations improved by about 2% from the baseline, a smaller improvement compared to the other two relations. Our Comparison performance is comparable with Ji et al. (2015)'s model without weak supervision.

Notably, the performance for Expansion dropped after domain adaptation (without extensions) by about 5%. We suspect that this is because the distributions of Expansion among other relations are very different (33% for explicit and 53% for implicit, c.f. Table 1). By applying Spectral Normalization, the performance improved and surpassed Explicit → Implicit.

Component-wise, Spectral Normalization helps two of the four relations (Contingency and Expansion), but hurts the performance of Comparison.

[2]The weakly labeled data includes sentences extracted from 1000 CNN articles, with explicit discourse connectives but without annotated discourse relations.

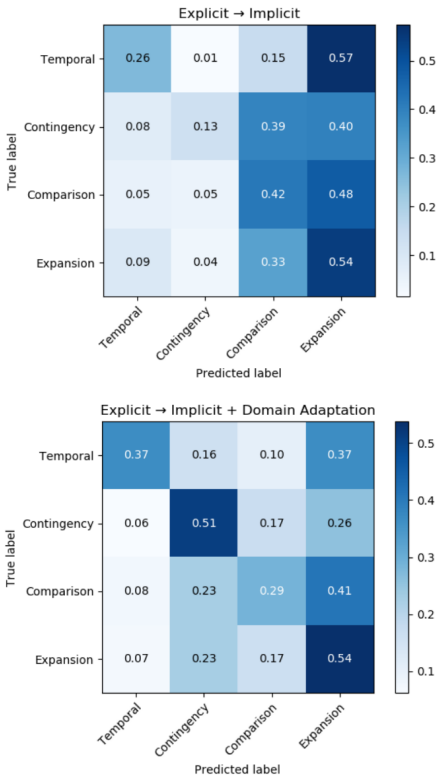


Figure 4: Normalized confusion matrices before and after unsupervised domain adaptation.

Label smoothing improves performance for all relations except Expansion; applying the reconstruction loss improves performance for all relations except Temporal. Overall, the best result on this task is to incorporate all components.

Error analysis Figure 4 shows the normalized confusion matrices before and after unsupervised domain adaptation. Before adaptation, Temporal and Contingency relations are often misclassified as Expansion, which is substantially improved after adaptation. The improvement in F score for Comparison is milder due to lower precision and higher recall, which is also reflected in the matrices. Finally, the drop of performance in Expansion adaptation can be traced through increased confusion between Expansion and Contingency.

5 How about a little supervision?

We have so far presented an unsupervised domain adaptation system that is not trained on any labels Y_t in the target domain. However, it is sometimes feasible to have some seed annotation that can be used to improve prediction. Hence we extend the model with an optional supervised component. We evaluate this extension by gradually adding labeled examples of implicit discourse relations, simulating situations when different numbers of labeled examples are available.

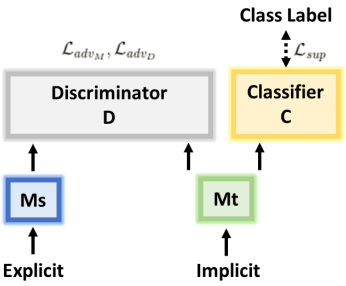


Figure 5: An extension component to incorporate supervision with our unsupervised adversarial domain adaptor.

5.1 Incorporating supervision

We extend the model with a supervised component, where a subset $X_t^L \subseteq X_t$ has labels Y_t^L. Illustrated in Figure 5, we jointly optimize the target encoder M_t and the classifier C according to an additional supervised loss:

$$\min_{M_t,C} \mathcal{L}_{sup}(X_t^L, Y_t^L) = \mathbb{E}_{(x_t^L, y_t^L)} - \sum_k \mathbb{1}[k = y_t^L] \log C(M_t(x_t^L)) \quad (9)$$

Effectively, we encourage the target encoder to jointly extract more discriminative features for all target examples (X_t), and learning target domain representations close to the source.

The full objective incorporates supervision of the in-domain labels by adding $\mathcal{L}_{sup}(X_t^L, Y_t^L)$ to the unsupervised objective:

$$\mathcal{L}^{sup}(X_s, Y_s, X_t, M_s, M_t, D) = \mathcal{L}^{unsup}(X_s, Y_s, X_t, M_s, M_t, D) + \min_{M_t,C} \mathcal{L}_{sup}(X_t^L, Y_t^L) \quad (10)$$

5.2 Data and settings

We synthesize the labeled target subset (X_t^L, Y_t^L) $(X_t^L \subseteq X_t)$ by randomly extracting subsets from the implicit training set and get their labels. The sizes of this subset range from 1382 to 13824 with a stepsize of 1382. Note that we use the entire implicit training set (X_t without relation labels Y_t) in the adversarial adaptation process as unlabeled data in the target domain, and the sampled labeled data is used in the supervised component only.

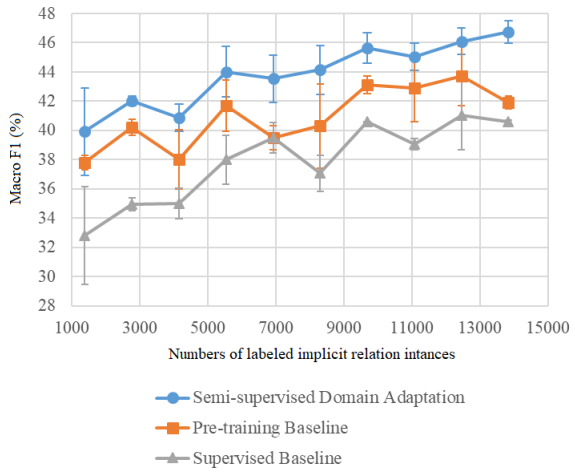


Figure 6: The average F1 (%) with varying numbers of labeled implicit relation training data.

We use the same hyper-parameters as the unsupervised experiment, except that we tune the learning rate on the implicit development set.

5.3 Systems

We compare three settings:

Supervised baseline The encoder and classifier trained on the sampled implicit instances (X_t^L, Y_t^L), optimizing Eq.(9).

Pre-training baseline Our model with the supervised component, but without the domain adaptation component. This setting is equivalent to pre-training on the explicit instances then fine-tuning on the sampled implicit instances. This is trained on the explicit training set (X_s, Y_s), plus the sampled implicit instances (X_t^L, Y_t^L), optimizing Eq.(5) and Eq.(9).

Semi-supervised domain adaptation Our full model with both the supervised and adaptation components, optimizing Eq.(10). The supervised component uses the sampled implicit instances (X_t^L, Y_t^L) for training.

5.4 Results

Since the added training data is randomly sampled, we average the performance across 3 different runs. Figure 6 shows the average F1 measure (y-axis) of the above three supervised systems, with varying numbers of labeled implicit relation training data (x-axis). Standard errors are also shown in the graph.

Our full system outperforms both the supervised baseline and the pre-training baseline, regardless of the amount of labeled target data. This evaluation also reveals that the pre-training baseline also improves upon the supervised baseline across the board, which means that the performance of implicit relation classification can be improved with pre-training on explicit relations.

Finally, the macro F1 of our system using full supervision is 47.50. Since we focus on domain adaptation and used very simple encoders, we do not attempt to achieve state-of-the-art (e.g., Dai and Huang (2018), Bai and Zhao (2018)). However this performance is on-par with many recent work using multi-task or GANs, including Lan et al. (2017) (47.80), Qin et al. (2017) (44.38, Reproduced results on four-way classification), and Liu et al. (2016) (44.98). These results confirm that our framework generalizes well with respect to the amount of supervision in the target domain.

6 Conclusion

Our work tackles implicit discourse relation classification in a low resource setting that is flexible to the amount of supervision. We present a new system based on the adversarial discriminative domain adaptation framework (Tzeng et al., 2017) for unsupervised domain adaptation from explicit discourse relation to implicit discourse relation. We propose a reconstruction loss to preserve the discriminability of features during adaptation, and we generalize the framework to make use of possibly available seed data by jointly optimizing it with a supervised loss. Our system outperforms prior work and strong adversarial baselines on unsupervised domain adaptation, and works effectively with varying amount of supervision.

Acknowledgments

We thank Ray Mooney, Eric Holgate, and the anonymous reviewers for their helpful feedback. This work was partially supported by the NSF Grant IIS-1850153.

References

Amal Al-Saif and Katja Markert. 2010. The leeds arabic discourse treebank: Annotating discourse connectives for arabic. In *LREC*.

Hongxiao Bai and Hai Zhao. 2018. Deep enhanced representation for implicit discourse relation recognition. In *COLING*.

Or Biran and Kathleen McKeown. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *ACL*.

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. 2017. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*.

Chloé Braud and Pascal Denis. 2014. Combining natural and artificial examples to improve implicit discourse relation identification. In *COLING*.

Chloé Braud and Pascal Denis. 2016. Learning connective-based word representations for implicit discourse relation identification. In *EMNLP*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Zeyu Dai and Ruihong Huang. 2018. Improving implicit discourse relation classification by modeling inter-dependencies of discourse units in a paragraph. In *NAACL*.

Lisheng Fu, Thien Huu Nguyen, Bonan Min, and Ralph Grishman. 2017. Domain adaptation for relation extraction with domain adversarial neural network. In *IJCNLP*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for twitter with adversarial neural networks. In *EMNLP*.

Yangfeng Ji, Gongbo Zhang, and Jacob Eisenstein. 2015. Closing the gap: Domain adaptation from explicit to implicit discourse relations. In *EMNLP*.

Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. 2017. Cross-language learning with adversarial neural networks. In *CoNLL*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Man Lan, Jianxiang Wang, Yuanbin Wu, Zheng-Yu Niu, and Haifeng Wang. 2017. Multi-task attention-based neural networks for implicit discourse relationship representation and identification. In *EMNLP*.

Man Lan, Yu Xu, and Zhengyu Niu. 2013. Leveraging synthetic discourse data via multi-task learning for implicit discourse relation recognition. In *ACL*.

Ming-Yu Liu and Oncel Tuzel. 2016. Coupled generative adversarial networks. In *NIPS*.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *AAAI*.

Daniel Marcu and Abdessamad Echihabi. 2002. An unsupervised approach to recognizing discourse relations. In *ACL*.

Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2004. The Penn Discourse Treebank. In *LREC*.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. In *ICLR*.

Umangi Oza, Rashmi Prasad, Sudheer Kolachina, Dipti Misra Sharma, and Aravind Joshi. 2009. The Hindi Discourse Relation Bank. In *The Third Linguistic Annotation Workshop*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Lucie Poláková, Jiří Mírovskỳ, Anna Nedoluzhko, Pavlína Jínová, Šárka Zikánová, and Eva Hajičová. 2013. Introducing the prague discourse treebank 1.0. In *IJCNLP*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *LREC*.

Rashmi Prasad, Bonnie Webber, and Aravind Joshi. 2014. Reflections on the Penn Discourse Treebank, comparable corpora, and complementary annotation. *Computational Linguistics*, 40(4):921–950.

Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. 2017. Adversarial connective-exploiting networks for implicit discourse relation classification. In *ACL*.

Paolo Russo, Fabio Maria Carlucci, Tatiana Tommasi, and Barbara Caputo. 2018. From source to target and back: symmetric bi-directional adaptive gan. In *CVPR*.

Attapol Rutherford and Nianwen Xue. 2015. Improving the inference of implicit discourse relations via classifying explicit discourse connectives. In *NAACL*.

Swami Sankaranarayanan, Yogesh Balaji, Carlos D. Castillo, and Rama Chellappa. 2018. Generate to adapt: Aligning domains using generative adversarial networks. In *CVPR*.

Wei Shi, Frances Yung, Raphael Rubino, and Vera Demberg. 2017. Using explicit discourse connectives in translation for implicit discourse relation classification. In *IJCNLP*.

Caroline Sporleder and Alex Lascarides. 2008. Using automatically labelled examples to classify rhetorical relations: An assessment. *Natural Language Engineering*, 14(3):369–416.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. 2015. Simultaneous deep transfer across domains and tasks. In *ICCV*.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *CVPR*.

Changxing Wu, Xiaodong Shi, Yidong Chen, Yanzhou Huang, and Jinsong Su. 2016. Bilingually-constrained synthetic data for implicit discourse relation recognition. In *EMNLP*.

Changxing Wu, Xiaodong Shi, Yidong Chen, Jinsong Su, and Boli Wang. 2017. Improving implicit discourse relation recognition with discourse-specific word embeddings. In *ACL*.

Ruochen Xu and Yiming Yang. 2017. Cross-lingual distillation for text classification. In *ACL*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*.

Deniz Zeyrek, Işın Demirşahin, AB Sevdik-Çallı, and Ruket Çakıcı. 2013. Turkish discourse bank: Porting a discourse annotation style to a morphologically rich language. *Dialogue and Discourse*, 4(2):174–184.

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2017. Aspect-augmented adversarial networks for domain adaptation. *Transactions of the Association for Computational Linguistics*, 5:515–528.

Yuping Zhou and Nianwen Xue. 2015. The Chinese Discourse TreeBank: a Chinese corpus annotated with discourse relations. *Language Resources and Evaluation*, 49(2):397–431.

Zhi Min Zhou, Man Lan, Zheng Yu Niu, Yu Xu, and Jian Su. 2010a. The effects of discourse connectives prediction on implicit discourse relation recognition. In *SIGDIAL*.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010b. Predicting discourse connectives for implicit discourse relation recognition. In *COLING*.

Evidence Sentence Extraction for Machine Reading Comprehension

Hai Wang[1*] **Dian Yu**[2] **Kai Sun**[3*] **Jianshu Chen**[2]
Dong Yu[2] **David McAllester**[1] **Dan Roth**[4]
[1]Toyota Technological Institute at Chicago, Chicago, IL, USA
[2]Tencent AI Lab, Bellevue, WA, USA [3]Cornell, Ithaca, NY, USA
[4]University of Pennsylvania, Philadelphia, PA, USA
{haiwang,mcallester}@ttic.edu, ks985@cornell.edu,
{yudian,jianshuchen,dyu}@tencent.com, danroth@seas.upenn.edu

Abstract

Remarkable success has been achieved in the last few years on some limited machine reading comprehension (MRC) tasks. However, it is still difficult to interpret the predictions of existing MRC models. In this paper, we focus on extracting evidence sentences that can explain or support the answers of multiple-choice MRC tasks, where the majority of answer options cannot be directly extracted from reference documents.

Due to the lack of ground truth evidence sentence labels in most cases, we apply distant supervision to generate imperfect labels and then use them to train an evidence sentence extractor. To denoise the noisy labels, we apply a recently proposed deep probabilistic logic learning framework to incorporate both sentence-level and cross-sentence linguistic indicators for indirect supervision. We feed the extracted evidence sentences into existing MRC models and evaluate the end-to-end performance on three challenging multiple-choice MRC datasets: MultiRC, RACE, and DREAM, achieving comparable or better performance than the same models that take as input the full reference document. To the best of our knowledge, this is the first work extracting evidence sentences for multiple-choice MRC.

1 Introduction

Recently, there have been increased interests in machine reading comprehension (MRC). In this work, we mainly focus on multiple-choice MRC (Richardson et al., 2013; Mostafazadeh et al., 2016; Ostermann et al., 2018): given a document and a question, the task aims to select the correct answer option(s) from a small number of answer options associated with this question. Compared to extractive and abstractive MRC tasks (e.g., (Rajpurkar et al., 2016; Kočiskỳ et al., 2018; Reddy et al., 2019)) where most questions can be answered using spans from the reference documents, the majority of answer options cannot be directly extracted from the given texts.

Existing multiple-choice MRC models (Wang et al., 2018b; Radford et al., 2018) take as input the entire reference document and seldom offer any explanation, making interpreting their predictions extremely difficult. It is a natural choice for human readers to use sentences from a given text to explain why they select a certain answer option in reading tests (Bax, 2013). In this paper, as a preliminary attempt, we focus on exacting ***evidence sentences*** that entail or support a question-answer pair from the given reference document.

For extractive MRC tasks, information retrieval techniques can be very strong baselines to extract sentences that contain questions and their answers when questions provide sufficient information, and most questions are factoid and answerable from the content of a single sentence (Lin et al., 2018; Min et al., 2018). However, we face unique challenges to extract evidence sentences for multiple-choice MRC tasks. The correct answer options of a significant number of questions (e.g., 87% questions in RACE (Lai et al., 2017; Sun et al., 2019)) are not extractive, which may require advanced reading skills such as inference over multiple sentences and utilization of prior knowledge (Lai et al., 2017; Khashabi et al., 2018; Ostermann et al., 2018). Besides, the existence of misleading wrong answer options also dramatically increases the difficulty of evidence sentence extraction, especially when a question provides insufficient information. For example, in Figure 1, given the reference document and question *"Which of the following statements is true according to the passage?"*, almost all the tokens in

* This work was done when H. W. and K. S. were at Tencent AI Lab, Bellevue, WA.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 696–707
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

the wrong answer option B *"In 1782, Harvard began to teach German."* appear in the document (i.e., sentence S_9 and S_{11}) while the question gives little useful information for locating answers. Furthermore, we notice that even humans sometimes have difficulty in finding pieces of evidence when the relationship between a question and its correct answer option is implicitly indicated in the document (e.g., *"What is the main idea of this passage?"*). Considering these challenges, we argue that extracting evidence sentences for multiple-choice MRC is at least as difficult as that for extractive MRC or factoid question answering.

Given a question, its associated answer options, and a reference document, we propose a method to extract sentences that can potentially support or explain the (question, correct answer option) pair from the reference document. Due to the lack of ground truth evidence sentences in most multiple-choice MRC tasks, inspired by distant supervision, we first extract *silver standard* evidence sentences based on the lexical features of a question and its correct answer option (Section 2.2), then we use these noisy labels to train an evidence sentence extractor (Section 2.1). To denoise imperfect labels, we also manually design sentence-level and cross-sentence linguistic indicators such as *"adjacent sentences tend to have the same label"* and accommodate all the linguistic indicators with a recently proposed deep probabilistic logic learning framework (Wang and Poon, 2018) for indirect supervision (Section 2.3).

Previous extractive MRC and question answering studies (Min et al., 2018; Lin et al., 2018) indicate that a model should be able to achieve comparable end-to-end performance if it can accurately predict the evidence sentence(s). Inspired by the observation, to indirectly evaluate the quality of the extracted evidence sentences, we only keep the selected sentences as the new reference document for each instance and evaluate the performance of a machine reader (Wang et al., 2018b; Radford et al., 2018) on three challenging multiple-choice MRC datasets: MultiRC (Khashabi et al., 2018), RACE (Lai et al., 2017), and DREAM (Sun et al., 2019). Experimental results show that we can achieve comparable or better performance than the same reader that considers the full context. The comparison between ground truth evidence sentences and automatically selected sentences indicates that there is still room for improvement.

Our primary contributions are as follows: 1) to the best of our knowledge, this is the first work to extract evidence sentences for multiple-choice MRC; 2) we show that it may be a promising direction to leverage various sources of linguistic knowledge for denoising noisy evidence sentence labels. We hope our attempts and observations can encourage the research community to develop more explainable MRC models that simultaneously provide predictions and textual evidence.

2 Method

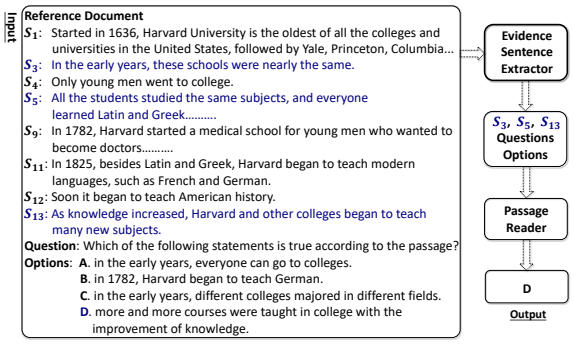


Figure 1: An overview of our pipeline. The input instance comes from RACE (Lai et al., 2017).

We will present our evidence sentence extractor (Section 2.1) trained on the noisy training data generated by distant supervision (Section 2.2) and denoised by an existing deep probabilistic logic framework that incorporates different kinds of linguistic indicators (Section 2.3). The extractor is followed by an independent neural reader for evaluation. See an overview in Figure 1.

2.1 Evidence Sentence Extractor

We use a multi-layer multi-head transformer (Vaswani et al., 2017) to extract evidence sentences. Let W_w and W_p be the word (subword) and position embeddings, respectively. Let M denote the total number of layers in the transformer. Then, the m-th layer hidden state h^m of a token is given by:

$$h^m = \begin{cases} W_w + W_p & \text{if } m = 0 \\ \text{TB}(h^{m-1}) & \text{if } 1 \leq m \leq M \end{cases} \quad (1)$$

where TB stands for the Transformer Block, which is a standard module that contains MLP, residual connections (He et al., 2016) and LayerNorm (Ba et al., 2016).

Compared to classical transformers, pre-trained transformers such as GPT (Radford et al., 2018)

and BERT (Devlin et al., 2018) capture rich world and linguistic knowledge from large-scale external corpora, and significant improvements are obtained by fine-tuning these pre-trained models on a variety of downstream tasks. We follow this promising direction by fine-tuning GPT (Radford et al., 2018) on a target task. Note that the pre-trained transformer in our pipeline can also be easily replaced with other pre-trained models, which however is not the focus of this paper.

We use (X, Y) to denote all training data, (X_i, Y_i) to denote each instance, where X_i is a token sequence, namely, $X_i = \{X_i^1, \ldots, X_i^t\}$ where t equals to the sequence length. For evidence sentence extraction, X_i contains one sentence in a document, a question, and all answer options associated with the question. Y_i indicates the probability that sentence in X_i is selected as an evidence sentence for this question, and $\sum_{i=1}^{N} Y_i = 1$, where N equals to the total number of sentences in a document. GPT takes as input X_i and produces the final hidden state h_i^M of the last token in X_i, which is further fed into a linear layer followed by a softmax layer to generate the probability:

$$P_i = \frac{\exp(W_y h_i^M)}{\sum_{1 \le i \le N} \exp(W_y h_i^M)} \quad (2)$$

where W_y is weight matrix of the output layer. We use Kullback-Leibler divergence loss $\mathrm{KL}(Y||P)$ as the training criterion.

We first apply distant supervision to generate noisy evidence sentence labels (Section 2.2). To denoise the labels, during the training stage, we use deep probabilistic logic learning (DPL) – a general framework for combining indirect supervision strategies by composing probabilistic logic with deep learning (Wang and Poon, 2018). Here we consider both sentence-level and cross-sentence linguistic indicators as indirect supervision strategies (Section 2.3).

As shown in Figure 2, during training, our evidence sentence extractor contains two components: a probabilistic graph containing various sources of indirect supervision used as a supervision module and a fine-tuned GPT used as a prediction module. The two components are connected via a set of latent variables indicating whether each sentence is an evidence sentence or not. We update the model by alternatively optimizing GPT and the probabilistic graph so that they reach an agreement on latent variables. After training, only the fine-tuned GPT is kept to make predictions for a new instance during testing. We provide more details in Appendix A and refer readers to Wang and Poon (2018) for how to apply DPL as a tool in a downstream task such as relation extraction.

2.2 Silver Standard Evidence Generation

Given correct answer options, we use a distant supervision method to generate the *silver standard* evidence sentences.

Inspired by Integer Linear Programming models (ILP) for summarization (Berg-Kirkpatrick et al., 2011; Boudin et al., 2015), we model evidence sentence extraction as a maximum coverage problem and define the value of a selected sentence set as the sum of the weights for the unique words it contains. Formally, let v_i denote the weight of word i, $v_i = 1$ if word i appears in the correct answer option, $v_i = 0.1$ if it appears in the question but not in the correct answer option, and $v_i = 0$ otherwise.[1]

We use binary variables c_i and s_j to indicate the presence of word i and sentence j in the selected sentence set, respectively. $\mathrm{Occ}_{i,j}$ is a binary variable indicating the occurrence of word i in sentence j, l_j denotes the length of sentence j, and L is the predefined maximum number of selected sentences. We formulate the ILP problem as:

$$\max \sum_i v_i c_i \quad s.t. \sum_j s_j \le L \quad (3)$$

$$s_j\ \mathrm{Occ}_{ij} \le c_i, \forall i, j \quad \sum_j s_j\ \mathrm{Occ}_{ij} \ge c_i, \forall i \quad (4)$$

$$c_i \in \{0, 1\}\ \forall i \quad s_j \in \{0, 1\}\ \forall j$$

2.3 Linguistic Indicators for Indirect Supervision

To denoise the imperfect labels generated by distant supervision (Section 2.2), as a preliminary attempt, we manually design a small number of sentence-level and cross-sentence linguistic indicators incorporated in DPL for indirect supervision. We briefly introduce them as follows and detail all indicators in Appendix A.3 and implementation details in Section 3.2.

We assume that a sentence is more likely to be an evidence sentence if the sentence and the question have similar meanings, lengths, coherent entity types, same sentiment polarity, or the

[1] We do not observe a significant improvement by tuning parameters v_i on the development set.

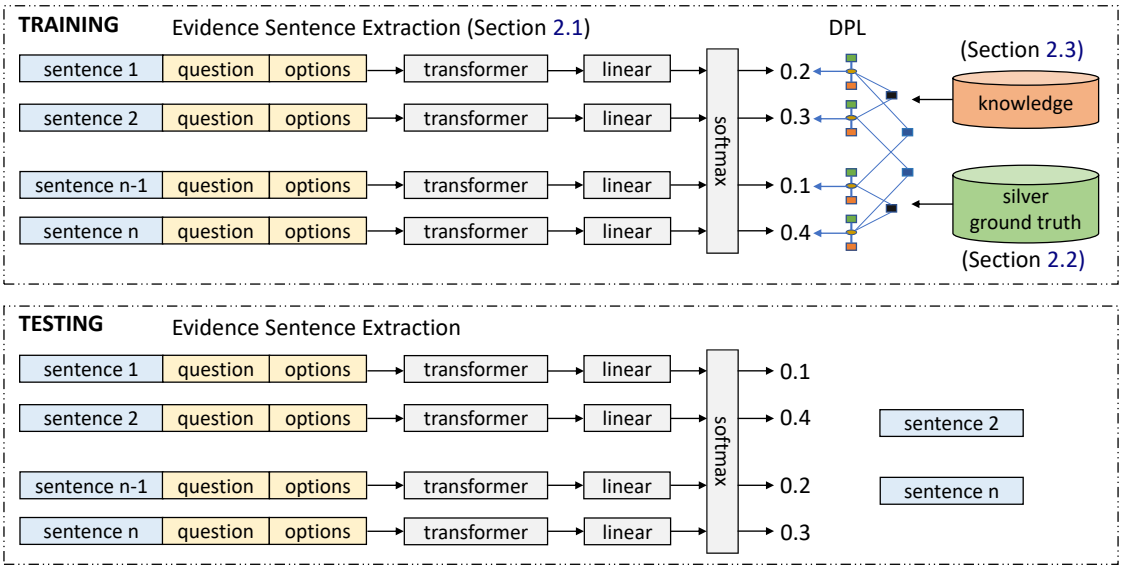


Figure 2: Deep probabilistic logic (DPL) framework for evidence sentence extraction. During testing, we only use trained evidence sentence extractor for prediction.

sentence is true (i.e., entailment) given the question. We assume that a good evidence sentence should be neither too long nor too short (i.e., $5 \leq$ # of tokens in sentence ≤ 40) considering informativeness and conciseness, and an evidence sentence is more likely to lead to the prediction of the correct answer option (referred as "reward"), which is motivated by our experiments that machine readers take as input the silver (or gold) standard evidence sentences achieve the best performance except for human performance on three multiple-choice machine reading comprehension datasets (Table 2, Table 3, and Table 4 in Section 3). We rely on both lexical features (e.g., lengths and entity types) and semantic features based on pre-trained word/paraphrase embeddings and external knowledge graphs to measure the similarity of meanings. We use existing models or resources for reward calculation, sentiment analysis and natural language inference.

For cross-sentence indicators, we consider that the same set of evidence sentences are less likely to support multiple questions and two evidence sentences that support the same question should be within a certain distance (i.e., evidence sentences for the same question should be within window size 8 (in sentences)). We also assume that adjacent sentences tend to have the same label. We will have more discussions about these assumptions in the data analysis (Section 3.6).

3 Experiments

3.1 Datasets

We use the following three latest multiple-choice machine reading comprehension datasets for evaluation. We show data statistics in Table 1.

MultiRC (Khashabi et al., 2018): MultiRC is a dataset in which questions can only be answered by considering information from multiple sentences. A question may have multiple correct answer options. Reference documents come from seven different domains such as elementary school science and travel guides. For each document, questions and their associated answer options are generated and verified by turkers.

RACE (Lai et al., 2017): RACE is a dataset collected from English language exams designed for middle (RACE-Middle) and high school (RACE-High) students in China, carefully designed by English instructors. The proportion of questions that requires reasoning is 59.2%.

DREAM (Sun et al., 2019): DREAM is a dataset collected from English exams for Chinese language learners. Each instance in DREAM contains a multi-turn multi-party dialogue, and the correct answer option must be inferred from the dialogue context. In particular, a large portion of questions require multi-sentence inference (84%) and/or commonsense knowledge (34%).

Dataset	# of documents			# of questions			Average # of sentences per document
	Train	Dev	Test	Train	Dev	Test	Train + Dev + Test
MultiRC	456	83	332	5,131	953	3,788	14.5 (Train + Dev)
DREAM	3,869	1,288	1,287	6,116	2,040	2,041	8.5
RACE	25,137	1,389	1,407	87,866	4,887	4,934	17.6

Table 1: Statistics of multiple-choice machine reading comprehension and question answering datasets.

3.2 Implementation Details

We use spaCy (Honnibal and Johnson, 2015) for tokenization and named entity tagging. We use the pre-trained transformer (i.e., GPT) released by Radford et al. (2018) with the same pre-processing procedure. When GPT is used as the neural reader, we set training epochs to 4, use eight P40 GPUs for experiments on RACE, and use one GPU for experiments on other datasets. When GPT is used as the evidence sentence extractor, we set batch size 1 per GPU and dropout rate 0.3. We keep other parameters default. Depending on the dataset, training the evidence sentence extractor generally takes several hours.

For DPL, we adopt the toolkit from Wang and Poon (2018). During training, we conduct message passing in DPL iteratively, which usually converges within 5 iterations. We use Vader (Gilbert, 2014) for sentiment analysis and ParaNMT-50M (Wieting and Gimpel, 2018) to calculate the paraphrase similarity between two sentences. We use the knowledge graphs (i.e., triples in ConceptNet v5.6 (Speer and Havasi, 2012; Speer et al., 2017)) to incorporate commonsense knowledge. To calculate the natural language inference probability, we first fine-tune the transformer (Radford et al., 2018) on several tasks, including SNLI (Bowman et al., 2015), SciTail (Khot et al., 2018), MultiNLI (Williams et al., 2018), and QNLI (Wang et al., 2018a).

To calculate the probability that each sentence leads to the correct answer option, we sample a subset of sentences and use them to replace the full context in each instance, and then we feed them into the transformer fine-tuned with instances with full context. If a particular combination of sentences $S = \{s_1, \ldots, s_n\}$ leads to the prediction of the correct answer option, we reward each sentence inside this set with $1/n$. To avoid the combinatorial explosion, we assume evidence sentences lie within window size 3. For another neural reader Co-Matching (Wang et al., 2018b), we use its default parameters. For DREAM and RACE, we set L, the maximum number of silver standard evidence sentences of a question, to 3. For MultiRC, we set L to 5 since many questions have more than 5 ground truth evidence sentences.

3.3 Evaluation on MultiRC

Since its test set is not publicly available, currently we only evaluate our model on the development set (Table 2). The fine-tuned transformer (GPT) baseline, which takes as input the full document, achieves an improvement of 2.2% in macro-average F1 ($F1_m$) over the previous highest score, 66.5%. If we train our evidence sentence extractor using the ground truth evidence sentences provided by turkers, we can obtain a much higher $F1_m$ 72.3%, even after we remove nearly 66% of sentences in average per document. We can regard this result as the supervised upper bound for our evidence sentence extractor. If we train the evidence sentence extractor with DPL as a supervision module, we get 70.5% in $F1_m$. The performance gap between 70.5% and 72.3% shows there is still room for improving denoising strategies.

3.4 Evaluation on RACE

As we cannot find any public implementations of recently published independent sentence selectors, we compare our evidence sentence extractor with InferSent released by Conneau et al. (2017) as previous work (Htut et al., 2018) has shown that it outperforms many state-of-the-art sophisticated sentence selectors on a range of tasks. We also investigate the ***portability*** of our evidence sentence extractor by combing it with two neural readers. Besides the fine-tuned GPT baseline, we use Co-Matching (Wang et al., 2018b), another state-of-the-art neural reader on the RACE dataset.

As shown in Table 3, by using the evidence sentences selected by InferSent, we suffer up to a 1.9% drop in accuracy with Co-Matching and up to a 4.2% drop with the fine-tuned GPT. In comparison, by using the sentences extracted by our sentence extractor, which is trained with DPL as a

Approach	$F1_m$	$F1_a$	EM_0
All-ones baseline (Khashabi et al., 2018)	61.0	59.9	0.8
Lucene world baseline (Khashabi et al., 2018)	61.8	59.2	1.4
Lucene paragraphs baseline (Khashabi et al., 2018)	64.3	60.0	7.5
Logistic regression (Khashabi et al., 2018)	66.5	63.2	11.8
Full context + Fine-Tuned Transformer (GPT, Radford et al. (2018))	68.7	66.7	11.0
Random 5 sentences + GPT	65.3	63.1	7.2
Top 5 sentences by ESE_{DS} + GPT	70.2	**68.6**	12.7
Top 5 sentences by ESE_{DPL} + GPT	**70.5**	67.8	**13.3**
Top 5 sentences by ESE_{gt} + GPT	**72.3**	**70.1**	**19.2**
Ground truth evidence sentences + GPT	78.1	74.0	28.6
Human Performance (Khashabi et al., 2018)	86.4	83.8	56.6

Table 2: Performance of various settings on the MultiRC development set. We use the fine-tuned GPT as the evidence sentence extractor (ESE) and the neural reader (ESE_{DS}: ESE trained on the silver standard evidence sentences; ESE_{DPL}: ESE trained with DPL as a supervision module; ESE_{gt}: ESE trained using ground truth evidence sentences; $F1_m$ macro-average F1; $F1_a$: micro-average F1; EM_0: exact match).

Approach	Dev			Test		
	Middle	High	All	Middle	High	All
Sliding Window (Richardson et al., 2013; Lai et al., 2017)	-	-	-	37.3	30.4	32.2
Co-Matching (Wang et al., 2018b)	-	-	-	55.8	48.2	50.4
Full context + GPT (Radford et al., 2018)	-	-	-	62.9	57.4	59.0
Full context + GPT	55.6	56.5	56.0	57.5	56.5	56.8
Random 3 sentences + GPT	50.3	51.1	50.9	50.9	49.5	49.9
Top 3 sentences by InferSent (question) + Co-Matching	49.8	48.1	48.5	50.0	45.5	46.8
Top 3 sentences by InferSent (question + all options) + Co-Matching	52.6	49.2	50.1	52.6	46.8	48.5
Top 3 sentences by ESE_{DS} + Co-Matching	58.1	51.6	53.5	55.6	48.2	50.3
Top 3 sentences by ESE_{DPL} + Co-Matching	57.5	52.9	54.2	57.5	49.3	51.6
Top 3 sentences by InferSent (question) + GPT	55.0	54.7	54.8	54.6	53.4	53.7
Top 3 sentences by InferSent (question + all options) + GPT	59.2	54.6	55.9	57.2	53.8	54.8
Top 3 sentences by ESE_{DS} + GPT	62.5	57.7	59.1	64.1	55.4	58.0
Top 3 sentences by ESE_{DPL} + GPT	63.2	56.9	58.8	**64.3**	56.7	58.9
Top 3 sentences by ESE_{DS} + full context + GPT	63.4	58.6	60.0	63.7	57.7	59.5
Top 3 sentences by ESE_{DPL} + full context + GPT	64.2	58.5	60.2	62.4	**58.7**	**59.8**
Silver standard evidence sentences + GPT	73.2	73.9	73.7	74.1	72.3	72.8
Amazon Turker Performance (Lai et al., 2017)	-	-	-	85.1	69.4	73.3
Ceiling Performance (Lai et al., 2017)	-	-	-	95.4	94.2	94.5

Table 3: Accuracy (%) of various settings on the RACE dataset. ESE_{DS}: evidence sentence extractor trained on the silver standard evidence sentences extracted from the rule-based distant supervision method.

supervision module, we observe a much smaller decrease (0.1%) in accuracy with the fine-tuned GPT baseline, and we slightly improve the accuracy with the Co-Matching baseline. For questions in RACE, introducing the content of answer options as additional information for evidence sentence extraction can narrow the accuracy gap, which might be due to the fact that many questions are less informative (Xu et al., 2018). Note that all these results are compared with 59% reported from Radford et al. (2018), if compared with our own replication (56.8%), sentence extractor trained with either DPL or distant supervision leads to gain up to 2.1%.

Since the problems in RACE are designed for human participants that require advanced reading comprehension skills such as the utilization of external world knowledge and in-depth reasoning, even human annotators sometimes have difficulties in locating evidence sentences (Section 3.6). Therefore, ***a limited number of evidence sentences might be insufficient for answering challenging questions***. Instead of removing "non-relevant" sentences, we keep all the sentences in a document while adding a special token before and after the extracted evidence sentences. With DPL as a supervision module, we see an improvement in accuracy of 0.9% (from 58.9% to 59.8%).

For our current supervised upper bound (i.e., assuming we know the correct answer option, we find the silver evidence sentences from rule-based distant supervision and then feed them into the fine-tuned transformer, we get 72.8% in accuracy, which is quite close to the performance of Amazon Turkers. However, it is still much lower than the ceiling performance. To answer questions that require external knowledge, ***it might be a promising direction to retrieve evidence sentences from external resources***, compared to only considering sentences within a reference document for multiple-choice machine reading comprehension tasks.

3.5 Evaluation on DREAM

See Table 4 for results on the DREAM dataset. The fine-tuned GPT baseline, which taks as input the full document, achieves 55.1% in accuracy on the test set. If we train our evidence sentence extractor with DPL as a supervision module and feed the extracted evidence sentences to the fine-tuned GPT, we get test accuracy 57.7%. Similarly, if we train the evidence sentence extractor only with silver standard evidence sentences extracted from the rule-based distant supervision method, we obtain test accuracy 56.3%, i.e., 1.4% lower than that with full supervision. Experiments demonstrate the effectiveness of our evidence sentence extractor with denoising strategy, and the usefulness of evidence sentences for dialogue-based machine reading comprehension tasks in which reference documents are less formal compared to those in RACE and MultiRC.

Approach	Dev	Test
Full context + GPT† (Sun et al., 2019)	55.9	55.5
Full context + GPT	55.1	55.1
Top 3 sentences by $ESE_{silver\text{-}gt}$ + GPT	50.1	50.4
Top 3 sentences by ESE_{DS} + GPT	55.1	56.3
Top 3 sentences by ESE_{DPL} + GPT	57.3	**57.7**
Silver standard evidence sentences + GPT	60.5	59.8
Human Performance†	93.9	95.5

Table 4: Performance in accuracy (%) on the DREAM dataset (Results marked with † are taken from Sun et al. (2019); $ESE_{silver\text{-}gt}$: ESE trained using silver standard evidence sentences).

3.6 Human Evaluation

Extracted evidence sentences, which help neural readers to find correct answers, may still fail to convince human readers. Thus we evaluate the quality of extracted evidence sentences based on human annotations (Table 5).

Dataset	Silver Sentences	Sentences by ESE_{DPL}
RACE-M	59.9	57.5
MultiRC	53.0	60.8

Table 5: Macro-average F1 compared with human annotated evidence sentences on the dev set (silver sentences: evidence sentences extracted by ILP (Section 2.2); sentences by ESE_{DPL}: evidence sentences extracted by ESE trained on silver stand ground truth, GT: ground truth evidence sentences).

MultiRC: Even trained using the noisy labels, we achieve a macro-average F1 score 60.8% on MultiRC, indicating the learning and generalization capabilities of our evidence sentence extractor, compared to 53.0%, achieved by using the noisy silver standard evidence sentences guided by correct answer options.

RACE: Since RACE does not provide the ground truth evidence sentences, to get the ground truth evidence sentences, two human annotators annotate 500 questions from the RACE-Middle development set.[2] The Cohen's kappa coefficient between two annotations is 0.87. For negation questions which include negation words (e.g., *"Which statement is not true according to the passage?"*), we have two annotation strategies: we can either find sentences that can directly imply the correct answer option; or the sentences that support the wrong answer options. During annotation, for each question, we use the strategy that leads to fewer evidence sentences.

We find that even humans have troubles in locating evidence sentences when the relationship between a question and its correct answer option is implicitly implied. For example, a significant number of questions require the understanding of the entire document (e.g., *"what's the best title of this passage"* and *"this passage mainly tells us that _"*) and/or external knowledge (e.g., *"the writer begins with the four questions in order to _"*, *"The passage is probably from _"* , and *"If the writer continues the article, he would most likely write about_"*). For 10.8% of total questions, at least one annotator leave the slot blank due to the challenges mentioned above. 65.2% of total questions contain at least two evidence sentences, and

[2]Annotations are available at https://github.com/nlpdata/evidence.

70.9% of these questions contain at least one adjacent sentence pair in their evidence sentences, which may provide evidence to support our assumption *adjacent sentences tend to have the same label* in Section 2.3.

The average and the maximum number of evidence sentences for the remaining questions is 2.1 and 8, respectively. The average number of evidence sentences in the full RACE dataset should be higher since questions in RACE-High are more difficult (Lai et al., 2017), and we ignore 10.8% of the total questions that require the understanding of the whole context.

3.7 Error Analysis

We analyze the predicted evidence sentences for instances in RACE for error analysis. Tough with a high macro-average recall (67.9%), it is likely that our method extracts sentences that support distractors. For example, to answer the question *"You lost your keys. You may call _"*, our system mistakenly extracts sentences *"Please call 5016666"* that support one of the distractors and adjacent to the correct evidence sentences *"Found a set of keys. Please call Jane at 5019999."* in the given document. We may need linguistic constraints or indicators to filter out irrelevant selected sentences instead of simply setting a hard length constraint such as 5 for all instances in a dataset.

Besides, it is possible that there is no clear sentence in the document for justifying the correctness of the correct answer. For example, to answer the question *"What does "figure out" mean ?"*, neither *"find out"* nor the correct answer option appears in the given document as this question mainly assesses the vocabulary acquisition of human readers. Therefore, all the extracted sentences (e.g., *"sometimes... sometimes I feel lonely, like I'm by myself with no one here."*, *"sometimes I feel excited, like I have some news I have to share!"*) by our methods are inappropriate. A possible solution is to predict whether a question is answerable following previous work (e.g., (Hu et al., 2019)) on addressing unanswerable questions in extractive machine reading comprehension tasks such as SQuAD (Rajpurkar et al., 2018) before to extract the evidence sentences for this question.

4 Related Work

4.1 Sentence Selection for Machine Reading Comprehension and Fact Verification

Previous studies investigate paragraph retrieval for factoid question answering (Chen et al., 2017; Wang et al., 2018c; Choi et al., 2017; Lin et al., 2018), sentence selection for machine reading comprehension (Hewlett et al., 2017; Min et al., 2018), and fact verification (Yin and Roth, 2018; Hanselowski et al., 2018). In these tasks, most of the factual questions/claims provide sufficient clues for identifying relevant sentences, thus often information retrieval combined with filters can serve as a very strong baseline. For example, in the FEVER dataset (Thorne et al., 2018), only 16.8% of claims require composition of multiple evidence sentences. For some of the cloze-style machine reading comprehension tasks such as CBT (Hill et al., 2016), Kaushik and Lipton (2018) demonstrate that for some models, comparable performance can be achieved by considering only the last sentence that usually contains the answer. Different from above work, we exploit information in answer options and use various indirect supervision to train our evidence sentence extractor, and previous work can actually be a regarded as a special case for our pipeline. Compared to Lin et al. (2018), we leverage rich linguistic knowledge for denoising imperfect labels.

Several work also investigate content selection at the token level (Yu et al., 2017; Seo et al., 2018), in which some tokens are automatically skipped by neural models. However, they do not utilize any linguistic knowledge, and a set of discontinuous tokens has limited explanation capability.

4.2 Machine Reading Comprehension with External Linguistic Knowledge

Linguistic knowledge such as coreference resolution, frame semantics, and discourse relations is widely used to improve machine comprehension (Wang et al., 2015; Sachan et al., 2015; Narasimhan and Barzilay, 2015; Sun et al., 2018) especially when there are only hundreds of documents available in a dataset such as MCTest (Richardson et al., 2013). Along with the creation of large-scale reading comprehension datasets, recent machine reading comprehension models rely on end-to-end neural models, and it primarily uses word embeddings as input. However, Wang et al. (2016); Dhingra et al. (2017,

2018) show that existing neural models do not fully take advantage of the linguistic knowledge, which is still valuable for machine reading comprehension. Besides widely used lexical features such as part-of-speech tags and named entity types (Wang et al., 2016; Liu et al., 2017; Dhingra et al., 2017, 2018), we consider more diverse types of external knowledge for performance improvements. Moreover, we accommodate external knowledge with probabilistic logic to potentially improve the interpretability of MRC models instead of using external knowledge as additional features.

4.3 Explainable Machine Reading Comprehension and Question Answering

To improve the interpretability of question answering, previous work utilize interpretable internal representations (Palangi et al., 2017) or reasoning networks that employ a hop-by-hop reasoning process dynamically (Zhou et al., 2018). A research line focuses on visualizing the whole derivation process from the natural language utterance to the final answer for question answering over knowledge bases (Abujabal et al., 2017) or scientific word algebra problems (Ling et al., 2017). Jansen et al. (2016) extract explanations that describe the inference needed for elementary science questions (e.g., *"What form of energy causes an ice cube to melt"*). In comparison, the derivation sequence is less apparent for open-domain questions, especially when they require external domain knowledge or multiple-sentence reasoning. To improve explainability, we can also check the attention map learned by neural readers (Wang et al., 2016), however, attention map is learned in end-to-end fashion, which is different from our work.

A similar work proposed by Sharp et al. (2017) also uses distant supervision to learn how to extract informative justifications. However, their experiments are primarily designed for factoid question answering, in which it is relatively easy to extract justifications since most questions are informative. In comparison, we focus on multi-choice MRC that requires deep understanding, and we pay particular attention to denoising strategies.

5 Conclusions

We focus on extracting evidence sentences for multiple-choice MRC tasks, which has not been studied before. We propose to apply distant supervision to noisy labels and apply a deep probabilistic logic framework that incorporates linguistic indicators for denoising noisy labels during training. To indirectly evaluate the quality of the extracted evidence sentences, we feed extracted evidence sentences as input to two existing neural readers. Experimental results show that we can achieve comparable or better performance on three multiple-choice MRC datasets, in comparison with the same readers taking as input the entire document. However, there still exist significant differences between the predicted sentences and ground truth sentences selected by humans, indicating the room for further improvements.

Acknowledgments

We thank the anonymous reviewers for their encouraging and helpful feedback.

References

Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. 2017. QUINT: Interpretable question answering over knowledge bases. In *Proceedings of the EMNLP (System Demonstrations)*, pages 61–66, Copenhagen, Denmark.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint*, stat.ML/1607.06450v1.

Stephen Bax. 2013. The cognitive processing of candidates during reading tests: Evidence from eye-tracking. *Language Testing*, 30(4):441–465.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the ACL*, pages 481–490, Portland, OR.

Lidong Bing, Sneha Chaudhari, Richard Wang, and William Cohen. 2015. Improving distant supervision for information extraction using label propagation through lists. In *Proceedings of the EMNLP*, pages 524–529, Lisbon, Portugal.

Florian Boudin, Hugo Mougard, and Benoit Favre. 2015. Concept-based summarization using integer linear programming: From concept pruning to multiple optimal solutions. In *Proceedings of the EMNLP*, pages 17–21, Lisbon, Portugal.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the EMNLP*, pages 632–642, Lisbon, Portuga.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the ACL*, pages 1870–1879, Vancouver, Canada.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the ACL*, pages 209–220, Vancouver, Canada.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the EMNLP*, pages 670–680, Copenhagen, Denmark.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the NAACL*, pages 4171–4186, Minneapolis, MN.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2018. Neural models for reasoning over multiple mentions using coreference. In *Proceedings of the NAACL-HLT*, pages 42–48, New Orleans, LA.

Bhuwan Dhingra, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2017. Linguistic knowledge as memory for recurrent neural networks. *arXiv preprint*, cs.CL/arXiv:1703.02620v1.

CJ Hutto Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the ICWSM*, pages 216–225, Qubec, Canada.

Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. Ukp-athene: Multi-sentence textual entailment for claim verification. *arXiv preprint*, cs.IR/1809.01479v2.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the CVPR*, pages 770–778, Las Vegas, NV.

Daniel Hewlett, Llion Jones, Alexandre Lacoste, et al. 2017. Accurate supervised and semi-supervised machine reading for long documents. In *Proceedings of the EMNLP*, pages 2011–2020, Copenhagen, Denmark.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the ICLR*, Caribe Hilton, Puerto Rico.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the EMNLP*, pages 1373–1378, Lisbon, Portugal.

Phu Mon Htut, Samuel Bowman, and Kyunghyun Cho. 2018. Training a ranking function for open-domain question answering. In *Proceedings of the NAACL-HLT (Student Research Workshop)*, pages 120–127, New Orleans, LA.

Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Dongsheng Li. 2019. Read+ verify: Machine reading comprehension with unanswerable questions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6529–6537, Honolulu, HI.

Peter Jansen, Niranjan Balasubramanian, Mihai Surdeanu, and Peter Clark. 2016. What's in an explanation? Characterizing knowledge and inference requirements for elementary science exams. In *Proceedings of the COLING*, pages 2956–2965, Osaka, Japan.

Divyansh Kaushik and Zachary C Lipton. 2018. How much reading does reading comprehension require? a critical investigation of popular benchmarks. In *Proceedings of the EMNLP*, pages 5010–5015, Brussels, Belgium.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the NAACL-HLT*, pages 252–262, New Orleans, LA.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. In *Proceedings of the AAAI*, pages 5189–5197, New Orleans, LA.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gáabor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the EMNLP*, pages 785–794, Copenhagen, Denmark.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of the ACL*, pages 1736–1745, Melbourne, Australia.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the ACL*, pages 158–167, Vancouver, Canada.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2017. Stochastic answer networks for machine reading comprehension. In *Proceedings of the ACL*, pages 1694–1704, Melbourne, Australia.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *Proceedings of the ACL*, pages 1725–1735, Melbourne, Australia.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and evaluation framework for deeper understanding of commonsense stories. In *Proceedings of the NAACL-HLT*, pages 839–849, San Diego, CA.

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *Proceedings of the ACL*, pages 1253–1262, Beijing, China.

Simon Ostermann, Michael Roth, Ashutosh Modi, Stefan Thater, and Manfred Pinkal. 2018. SemEval-2018 Task 11: Machine comprehension using commonsense knowledge. In *Proceedings of the SemEval*, pages 747–757, New Orleans, LA.

Hamid Palangi, Paul Smolensky, Xiaodong He, and Li Deng. 2017. Question-answering with grammatically-interpretable representations. *arXiv preprint*, cs.CL/1705.08432v2.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. In *Preprint*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of th ACL*, pages 784–789, Melbourne, Australia.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the EMNLP*, pages 2383–2392, Austin, TX.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the EMNLP*, pages 193–203, Seattle, WA.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Mrinmaya Sachan, Kumar Dubey, Eric Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Proceedings of the ACL*, pages 239–249, Beijing, China.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2018. Neural speed reading via Skim-RNN. In *Proceedings of the ICLR*, New Orleans, LA.

Rebecca Sharp, Mihai Surdeanu, Peter Jansen, Marco A Valenzuela-Escárcega, Peter Clark, and Michael Hammond. 2017. Tell me why: Using question answering as distant supervision for answer justification. In *Proceedings of the CoNLL*, pages 69–79, Vancouver, Canada.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI*, pages 4444–4451, San Francisco, CA.

Robyn Speer and Catherine Havasi. 2012. Representing general relational knowledge in ConceptNet 5. In *Proceedings of the LREC*, pages 3679–3686, Istanbul, Turkey.

Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. DREAM: A challenge dataset and models for dialogue-based reading comprehension. *Transactions of the Association of Computational Linguistics*, 7:217–231.

Yawei Sun, Gong Cheng, and Yuzhong Qu. 2018. Reading comprehension with graph-based temporal-casual reasoning. In *Proceedings of the COLING*, pages 806–817, Santa Fe, NM.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the NAACL-HLT*, pages 809–819, New Orleans, LA.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the NIPS*, pages 5998–6008, Long Beach, CA.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint*, cs.CL/1804.07461v1.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the ACL*, pages 700–706, Beijing, China.

Hai Wang, Takeshi Onishi, Kevin Gimpel, and David McAllester. 2016. Emergent predication structure in hidden state vectors of neural readers. In *Proceedings of the Repl4NLP*, pages 26–36, Vancouver, Canada.

Hai Wang and Hoifung Poon. 2018. Deep probabilistic logic: A unifying framework for indirect supervision. In *Proceedings of the EMNLP*, pages 1891–1902, Brussels, Belgium.

Shuohang Wang, Mo Yu, Shiyu Chang, and Jing Jiang. 2018b. A co-matching model for multi-choice reading comprehension. In *Proceedings of the ACL*, pages 1–6, Melbourne, Australia.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018c. R^3: Reinforced reader-ranker for open-domain question answering. In *Proceedings of the AAAI*, pages 5981–5988, New Orleans, LA.

John Wieting and Kevin Gimpel. 2018. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the ACL*, pages 451–462, Melbourne, Australia.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the NAACL*, pages 1112–1122, New Orleans, LA.

Yichong Xu, Jingjing Liu, Jianfeng Gao, Yelong Shen, and Xiaodong Liu. 2018. Dynamic fusion networks for machine reading comprehension. *arXiv preprint*, cs.CL/1711.04964v2.

Wenpeng Yin and Dan Roth. 2018. TwoWingOS: A two-wing optimization strategy for evidential claim verification. In *Proceedings of the EMNLP*, pages 105–114, Brussels, Belgium.

Adams Wei Yu, Hongrae Lee, and Quoc Le. 2017. Learning to skim text. In *Proceedings of the ACL*, pages 1880–1890, Vancouver, Canada.

Mantong Zhou, Minlie Huang, and Xiaoyan Zhu. 2018. An interpretable reasoning network for multi-relation question answering. In *Proceedings of the COLING*, pages 2010–2022, Santa Fe, NM.

SIMVECS: Similarity-based Vectors for Utterance Representation in Conversational AI Systems

Ashraf Mahgoub
Purdue University / West Lafayette, IN
amahgoub@purdue.edu

Youssef Shahin
Microsoft / Redmond, WA
Youssef.Shahin@microsoft.com

Riham Mansour
Microsoft / Redmond, WA
rihamma@microsoft.com

Saurabh Bagchi
Purdue University / West Lafayette, IN
sbagchi@purdue.edu

Abstract

Conversational AI systems are gaining a lot of attention recently in both industrial and scientific domains, providing a natural way of interaction between customers and adaptive intelligent systems. A key requirement in these systems is the ability to efficiently parse user queries, understand the intent behind each query, and provide adequate responses to users. Therefore, many applications such as conversation bots and smart IoT devices has a natural language understanding (LU) service integrated within. One of the greatest challenges of language understanding services is efficient utterance (sentence) representation in vector space, which is an essential step for most ML tasks. In this paper, we propose a novel approach for generating vector space representations of conversational utterances using pair-wise similarity metrics. The proposed approach uses only a few corpora to tune the weights of the similarity metric without relying on external general purpose ontologies. Our experiments confirm that the generated vectors can improve the performance of LU services in unsupervised, semi-supervised and supervised learning tasks over state-of-the-art prior works.

1 Introduction

Challenges of conversational AI systems: Conversational AI systems empower virtual assistants in many applications such as conversation bots (also known as Chatbots) and smart IoT devices. Their ability to understand user spoken commands and identify the user's intent(s) is one of their main benefits. However, this is a very challenging task due to the diversity of domains and languages they are required to support. For example, Microsoft LUIS[1] currently supports 12 languages, whereas IBM Watson [2] supports 10 languages in their language understanding services. Moreover, conversational text queries are often very short and sparse, which hinders conventional text representations such as Bag-of-words (BOW) from capturing adequate features of the utterance.

In LU services, a language understanding application is often represented as a group of intents and entities that serve a specific business application. For example, a restaurant application may define intents such as *Order Food*, *Show Menu* and *Cancel Order*. The task of the language understanding service then is to classify newly received queries (utterances) into one (or more) of the defined intents.

Although many prior works were focused on constructing word-level vector representation such as (Mikolov et al., 2013) and (Pennington et al., 2014), generating an utterance-level vector representation is still a challenging task. Existing language modeling (LM) based approaches such as Para2Vec (Le and Mikolov, 2014) rely on deep neural networks to generate vector representations for the paragraph-level. However, these approaches are normally trained with an abundance of utterances to achieve the required performance, which is usually not present in the conversational text domain (Boyanov et al., 2017). Moreover, pre-trained vectors (i.e., vectors generated from an external corpus) provide the same vector representation for all domains (i.e. static). The desirable characteristic on the other hand is that for the same utterance to have different vector representations, and correspondingly, different intents, in different domains. Additionally, the current approaches make the embeddings more prone to bias towards the domain of the training data (Bolukbasi et al., 2016).

[1] www.luis.ai/home

[2] www.ibm.com/watson/

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 708–717
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

We propose a vector representation method, SIMVECS that generates dynamic utterance-level vector representations for different LU applications. With SIMVECS, each utterance is represented as a vector of similarity scores to a set of automatically identified "representative utterances" within the same application. This way the same utterance can have different representations depending on the application. As an example: the utterance *"I want a large pizza"* can be of type *Order Food* for a restaurant application, while the same utterance can be of type *None* (i.e., outlier) for a bus-tracking application. Therefore, application-determined vector representations are essential for capturing the semantics of utterances in LU services and hence accurate mapping to user-defined intents.

The following list represents our key contributions:

1. A novel approach to combine multiple similarity sub-metrics into one metric, automatically adjusting the weight for each sub-metric to the overall similarity score.

2. A novel approach for the vector representation of each utterance in a conversational AI system.

3. A semi-supervised algorithm for refining the vector representations based on user's feedback.

The rest of the paper is organized as follows. Section 2 covers related work and separates our approach from existing solutions. Section 3 gives an overview of the main components in our proposed solution. Section 4 describes how we calculate the similarity scores between utterances and generate the vector representations. Sections 5, 6, and 7 evaluate the generated vectors in unsupervised, semi-supervised, and supervised learning tasks respectively and compares the performance of SIMVECS to several baselines.

2 Related Work

2.1 Similarity metrics

Measuring the similarity between natural language sentences is crucial in many tasks such as: Question-answering systems (Achananuparp et al., 2008b) and information retrieval (Li et al., 2006; Zahran et al., 2015). Authors in (Achananuparp et al., 2008a) provide an evaluation for 14 different similarity metrics. The authors reported the best metric found was a composite between word-order and ontology-based similarity. However, the weights for how much each of the two sub-metrics contributes to the overall similarity metric is selected based on human intuition. Such selection becomes harder when several sub-metrics are considered and multiple domains have to be satisfied. We consider these weights as hyper-parameters and hence use an automated technique that applies Genetic-Algorithms (GA) (Mitchell, 1996) to find the optimal weights as it has been used in multiple hyper-parameter tuning tasks such as in (Friedrichs and Igel, 2005), (Lam et al., 2001), and (Mahgoub et al., 2017).

2.2 Vector Representation

One of the most common vector representation for both documents and sentences is the Bag-of-Words (BOW) model (Harris, 1954). In BOW, the sentence is represented as a binary vector that donates the existence or absence of individual words (i.e. vocabulary) in that sentence. One of the major disadvantages of BOW representation is the loss of word order. Consequently, two sentences with totally different meaning can have very close representation just because they use similar vocabulary. A variation of the model is bag-of-ngrams (Mcnamee and Mayfield, 2004), which aims at preserving the word order. However, both representations are very sparse and generate vectors with very high dimensions. (Mikolov et al., 2013) proposed a technique that uses deep neural-networks to learn efficient representations for the word level. Although the trained vectors capture many semantic features, generating a sentence-level representation from individual words vectors is still a challenging task. One simple approach is to represent the sentence as weighted average of all the words in the document. However, this approach has the same weakness of not preserving the word order (Le and Mikolov, 2014). A recent approach proposed by (Le and Mikolov, 2014) generates both word-level and paragraph-level representations. However, the approach relies on training the network with a large corpus with billions of tokens, which is rarely available in the conversational text domain. Moreover, the generated vectors can still suffer from being biased by the train-

ing data domain and cannot generalize for different domains. (Dai and Le, 2015) proposed an approach to improve the vector representations with pre-trained recurrent neural networks. The proposed approach showed significant improvement over both BOW and Paragraph vectors. SIMVECS uses only a few corpora to tune the weights of the similarity metric without relying on external general purpose ontologies.

3 Overview of SIMVECS

SIMVECS relies on pair-wise similarity metrics. Each metric serves as a function that assigns similarity (or distance) scores to a pair of utterances. Many similarity metrics have been proposed in the literature. Although we use only six, our solution is generic and can incorporate any additional similarity sub-metrics.

3.1 Similarity sub-metrics definition

The six similarity sub-metrics which SIMVECS uses are:

(1)Unigrams: measures similarity based on the inverse-document-frequency (IDF) scores of common unigrams. The resulting score is then normalized by dividing over the sum of IDF scores of all unique words in the two utterances:

$$Sim_{uni}(U_i.U_j) = \frac{\sum_{w \in U_i \cap U_j} IDF[w]}{\sum_{w' \in U_i \cup U_j} IDF[w']} \quad (1)$$

(2)Character N-grams: measures similarity based on the overlapping character n-gram tokenization. We use an equation similar to the unigram except that words (and their corresponding IDF scores) are replaced by overlapping character N-grams. We use tokens of size n=4 as recommended by literature (Mcnamee and Mayfield, 2004).

(3)Bigrams: similar to unigram except that it uses tokens of two adjacent words. For each bigram, we set the IDF score to be the max between the two words in the token (presented as IDF_{bi}).

(4)Trigrams: similar to bigrams except that it uses tokens of three adjacent words.

(5)Utterance Length: this captures the similarity between a pair of utterances based on their number of tokens. Although this might be a dangerous feature to rely on individually, it becomes very useful in the domain of conversational text when combined with other features. For example, Table. 1 shows the average utterance lengths per intent in the WebApp corpus. We observe a large variance in utterance lengths of different intents. This is because one might need more tokens to specify a complex intent such as booking a flight (which requires lots of details) than simpler intents like asking for help or canceling a request. We use the following formula for utterance length similarity:

$$Sim_{len}(U_i.U_j) = \frac{min(Length(U_i), Length(U_j))}{max(Length(U_i), Length(U_j))} \quad (2)$$

(6)Word order: measures the normalized difference of word order between the two utterances.

$$Sim_{wo}(U_i.U_j) = 1 - \frac{||ri - rj||}{||ri + rj||} \quad (3)$$

where ri and rj are word order vectors (a vector which represents the order of each word in the utterance) of utterances U_i and U_j respectively.

3.2 IDF scores calculation

The first 4 metrics (i.e. unigram, bigram, trigrams, and character N-grams) rely on pre-calculated IDF scores. All these IDF scores are pre-calculated from a large corpus of conversational text generated from Cortana virtual assistant[3]. This corpus contains 18M utterances that resembles conversations between a user and Cortana in different domains. With the calculated IDF scores, these sub-metrics can be calculated and used to calculate a composite similarity metric as follows:

$$Sim_{comp}(U_i.U_j, \bar{W}) = \bar{W} \quad . \quad \bar{Sim}$$
$$= [W1..W6] \begin{bmatrix} Sim_{uni}(U_i.U_j) \\ Sim_{char}(U_i.U_j) \\ Sim_{bi}(U_i.U_j) \\ Sim_{tri}(U_i.U_j) \\ Sim_{len}(U_i.U_j) \\ Sim_{wo}(U_i.U_j) \end{bmatrix} \quad (4)$$

where W_i's are normalized weights which represent the collaboration of each sub-similarity metric to the overall metric. W_i's serve as hyper-parameters that control the quality of the composite similarity metric. The different sub-metrics have different relative importances in detecting similarities between utterances within the application. Therefore, the weights should be tuned automatically according to the LU application.

[3]https://www.microsoft.com/en-us/cortana

3.3 Application-based weights tuning

In this section, we describe our method in tuning the weight (W_i) for every sub-metric in Eq. 4. First, we use 15 real-world applications from a popular language understanding service to serve as our training and testing data set. These applications represent different domains (e.g. restaurants, flight booking services, smart homes, etc.) and they are created by system admins. Therefore, they contain a user-defined label for every utterance, which serves as our ground truth. For every pair of utterances with the same label, we assign a similarity score of 1 (max similarity). Similarly, for every pair of utterances with different labels, we assign a similarity score of 0 (min similarity). Now the task of tuning the weights for the sub-metrics can be viewed as an optimization problem, which is given by the following formula:

$$\bar{W}^* = \underset{\bar{W}}{\operatorname{argmin}} \sum_{\forall i,j} RMSE(Sim_{gt}(U_i.U_j), Sim_{comp}(U_i.U_j, \bar{W})) \quad (5)$$

Where Sim_{gt} is the ground truth similarity (either 0 or 1), $RMSE$ is the root mean square error between the estimated similarity score and the ground truth. Therefore, the target of equation 5 is to find the vector of weights that minimizes the differences between the estimated similarity scores and the ground truth similarly provided by application users. We use genetic algorithms (GA) to find the best values of $\bar{W}^*$. GA is a metaheuristic optimization algorithm that is inspired by biological evolution. It has the nice feature of balancing between exploration (a.k.a mutation) and exploitation (a.k.a crossover) of different solution candidates (a.k.a chromosome) in the search space (Črepinšek et al., 2013). GA is favorable in solving optimization problems which convexity is not known, since it does not rely on derivative information in finding good search directions (i.e. derivative-free). We used all pair-wise utterances from the 15 applications to train and validate our approach. Each candidate solution in GA simply represents a vector of weights in Eq. 4, and the fitness function is the resulting sum or RMSEs across all pairs of utterances (the lower the better). We perform 5-fold cross validation on the 15 applications and take the average of the best vectors of each fold. The resulting vector is then used for all subsequent experiments.

Intents	Avg. Tokens	Std. Tokens
Change Password	8.625	1.4
Delete account	7.35	1.11
Download video	7	0
Export data	10.2	2.28

Table 1: Average number of tokens per utterance for the WebApp Corpora.

4 Similarity-based utterance representation

We estimate all pair-wise similarity scores using Eq. 4 and store them in a matrix of size NxN where N is the number of utterances. We then apply Principal Component Analysis (PCA) (Wold et al., 1987) to reduce the dimensionality of this matrix. This pair-wise similarity matrix, *SimMatrix* for short, is then used to generate the vector representation for each utterance. Consider the example shown in Fig. 1. On the left hand, we show 10 utterances (selected from (Coucke et al., 2018)) and their corresponding intents. On the right hand, we show the corresponding *SimMatrix* (before we apply PCA). The resulting matrix is a symmetric matrix with all its diagonal values = 1, representing maximum similarity. Then we use PCA to reduce the number of columns, loosely speaking, creating a set of representative utterances in a data-driven manner. At this point, we use each row as the vector representation of the corresponding utterance. Thus, each representative utterance serves as a dimension in the vector space, allowing utterances with similar neighbors to have similar vector representations.

This approach has a number of advantages over conventional vector representations (such as BOW) and LM based techniques (such as Para2Vec) in the domain of conversational text understanding. The data collected by (Braun et al., 2017) shows that conversational text tends to be very short with an average of 7.8 tokens per utterance. Moreover, 80% of the collected utterances are shorter than 9 tokens. This makes BOW representation very sparse. Also for LM-based techniques, it is hard to learn efficient vector representations because of the shortness of the context sequences used for training. Another advantage of SIMVECS is the easier detection of utterances that have no intent (i.e. the "None" intent utterances). As shown Fig. 1, the "None" intent utterances are expected to have similarity scores close to zero to

all other utterances, including other "None" utterances. This makes them located closer to the origin in SIMVECS's vector space and allows distance-based clustering techniques (such as K-means) to group "None" intent utterances in the same cluster. The second advantage is representative vector lengths: instead of using vectors of arbitrary lengths or length equal to the vocabulary size, we use vectors of length equal to the number of representative utterances in the given application. This can reduce the size of the generated vector representation significantly, especially when large vocabularies are used, while the LU application itself may have only a few tens or hundreds of utterances.

5 Unsupervised learning with conversational text

The problem of applying unsupervised learning techniques to text documents has been studied by many researchers in several domains such as (Beil et al., 2002),(Aggarwal and Zhai, 2012), and (Huang, 2008). The problem becomes more challenging with conversational text because of the shortness and the sparsity of the documents (Chen et al., 2011). We can categorize these techniques based on the input they need to perform clustering into two categories: 1) Techniques that require a vector representation for the data points, such as K-means and SVD. 2) Techniques that require a similarity (distance) function such as Affinity-Propagation (Frey and Dueck, 2007) and DBSCAN (Ester et al., 1996). However, the second category still requires an efficient vector representation to estimate the distances between pairs of utterances. We evaluate the efficacy of SIMVECS generated vectors in the unsupervised learning task against several baselines. We vary the vector representation while the clustering algorithm itself (K-means) remains the same. We show a comparison against the following techniques:

Spherical K-means (Buchta et al., 2012): This baseline uses BOW representation for the utterances and cosine-similarity as a distance function.

LDA (Blei et al., 2003): This baseline represents documents as probability distributions over latent topics. Documents with similar topic assignments are grouped together. Similar to K-means, it takes the number of latent topics (clusters) as an input. We set the number of topics to the number of intents in the corpus and then assign each utterance to the cluster with the maximum probability.

LDA + K-means: Here we use LDA's latent topic distribution as a vector representation to the utterance. Then we apply K-means for clustering the utterances.

Seq-Auto-encoder + K-means (Dai and Le, 2015): A Neural-network based technique using recurrent language models. The network is trained on a large corpus of conversational text generated from the same Cortana corpus used for generating the IDF scores. Afterwards, the network is used to encode each utterance of the test data in a vector of length 1024. K-means is then used to cluster the utterances.

Fast-text + K-means (Joulin et al., 2016): Also a Neural-network based technique that incorporates several features such as BOW and N-gram features. The model generates word-level vectors which are averaged together to form sentence-level representations. We used English pre-trained word vectors[4]. These vectors are pre-trained with 16 billion tokens collected from *Wikipedia 2017, UMBC webbase* corpus and *statmt.org* news.

SIMVECS+ K-means: K-means applied to our similarity-based vectors.

To make the comparison fair, K-means algorithm with the same number of clusters (K*) is used for the all techniques (Except LDA). Here K* is the number of intents in the corpora. For LDA, we set number of latent topics to be also K*.

Corpora	# Intents	# Utterances	# Tokens
AskUbuntu	5	162	1289
Chatbot	2	200	1539
WebApp	8	89	717
Combined	15	451	3545

Table 2: Conversational text corpora details.

5.1 Dataset description

We use the conversational text dataset presented in (Braun et al., 2017)[5]. The dataset represents a collection of three corpora, two corpora were extracted from StackExchange (Ask Ubuntu & WebApp), while the third one was extracted from a Telegram chatbot. *Combined* is a corpus that combines the intents of all three. Table 2 shows the number of intents, number of utterances, and number of tokens for each corpus in the dataset.

[4]https://fasttext.cc/docs/en/english-vectors.html

[5]The dataset is publicly available and can be obtained here: https://github.com/sebischair/NLU-Evaluation-Corpora

Utterance ID	Intent	Utterance
U1	AddToPlaylist	add the current tune, to my, Rock Gaming, playlist
U2	AddToPlaylist	add villotta, to The MetalSucks Playlist, playlist
U3	SearchCreative Work	Please look up the Atheist Manifesto: The Case Against Christianity, album, .
U4	SearchCreative Work	Please look up the painting, Beyond Iconic: Photographer Dennis Stock, .
U5	BookRestaurant	book a spot for 8, at The Kitchin, on october the 13th, 2039,
U6	BookRestaurant	Book a reservation for 8, at a restaurant, that serves chicken fried bacon, in Aruba,
U7	RateBook	Give the current, book, im reading zero, points, out of 6,
U8	RateBook	rate the current, book, three, out of 6,
U9	None	PayPal
U10	None	OkCupid

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8	U_9	U_{10}
U_1	1	0.3	0.03	0.03	0.03	0.02	0.1	0.12	0	0
U_2	0.3	1	0.03	0.03	0.03	0.01	0.03	0.04	0	0
U_3	0.03	0.03	1	0.23	0.03	0.01	0.03	0.03	0	0
U_4	0.03	0.03	0.23	1	0.03	0.01	0.03	0.03	0	0
U_5	0.03	0.03	0.03	0.03	1	0.19	0.08	0.1	0	0
U_6	0.02	0.01	0.01	0.01	0.19	1	0.06	0.07	0	0
U_7	0.1	0.03	0.03	0.03	0.08	0.06	1	0.41	0	0
U_8	0.12	0.04	0.03	0.03	0.1	0.07	0.41	1	0	0
U_9	0	0	0	0	0	0	0	0	1	0.01
U_{10}	0	0	0	0	0	0	0	0	0.01	1

Figure 1: An example showing extracting the vector representation for each utterance. The table on the left shows 10 utterances (examples) for 5 different intents, whereas the table on the right shows the corresponding SimMatrix for the 10 utterances. Each row (or column) is then used as the vector representation for the corresponding utterance

5.2 Unsupervised learning results

In this section, we show the efficacy of SIMVECS's vector representation and compare to other baseline techniques. We collected both purity and normalized mutual information (NMI) scores for the generated clusters. We omit NMI scores for space reasons as it shows the same trend as purity. As shown in table 3, SIMVECS shows better performance (in terms of Purity) in comparison to other baseline except in one corpors, "Chatbot". Our solution (SIMVECS+ K-means) outperforms all baselines by at least 10% on average. Fast-text, Seq-Auto-Encoder, and BOW have very similar performance. We also notice that LDA+K-means outperforms LDA by 9%, suggesting that using topic distributions as vector representations is more efficient than mapping the utterance to the topic with the maximum corresponding probability. We also notice that Fast-text outperforms all other techniques (including SIMVECS) in one corpus (Chatbot). The reason is that Fast-text through training on billions of words, can find similarities between words that belong to the same domain. For example: the vector representation for the words "Pizza" and "Burger" have a cosine-similarity of 0.63 using Fast-text vectors as both words are very frequent in the "Restaurants" domain. Although this might be useful in some situations, it can be very misleading in others, causing non-similar utterances to have high silimarity scores. For instance, a particular restaurant might only serve burgers while any pizza orders are not supported and therefore should be considered outliers (i.e. "None" intent). This means the similarity between the two words will cause an overlap between these two intents ("Order" and "None"), driving the clustering algorithm to mistakenly combine the two intents into one cluster. This behavior is highlighted in the poor performance of Fast-text in all other corpora compared to SIMVECS.

6 Semi-Supervised learning with conversational text

In this section, we evaluate the efficacy of SIMVECS vector representations in semi-supervised learning tasks. One of the main requirements in language understanding services is to improve their performance during operation using active learning techniques. This is achieved by adapting to different user perspectives in different domains. Moreover, it is hard in many cases to perform clustering efficiently without taking the user perspective into consideration. For instance, Fig. 2 shows an example of two groups of utterances that can be clustered at different levels of granularity based on the user's needs. In many such cases, the same set of utterances can be clustered in different ways according to different

Corpora	Combined	AskUbuntu	Chatbot	WebApplications	Average
Spherical Kmeans	61%	63%	61%	64%	62%
LDA	46%	59%	61%	35%	50%
LDA+Kmeans	55%	67%	62%	54%	59%
Seq-Auto-encoder+Kmeans	65%	64%	62%	57%	62%
Fast-Text+Kmeans	53%	56%	**94%**	49%	63%
SIMVECS+Kmeans	**71%**	**83%**	61%	**76%**	**73%**

Table 3: Purity scores for SIMVECS vs several baselines

criteria. Moreover, it is not known which criteria is to be used by the clustering algorithm without user feedback. Therefore, semi-supervised learning is typically used in directing SIMVECS to the correct context-sensitive clustering. In this section, we evaluate the ability of SIMVECS in a semi-supervised learning mode, compared to several baselines.

In semi-supervised clustering, few data points are labeled and used as constraints to the clustering technique. These constraints are of the form **"Must-Link"** and **"Cannot-Link"** for pairs of data points. A "Must-Link" constraint arises when the user indicates that two utterances belong to the same intent, while a "Cannot-Link" constraint is when to the user indicates two utterances as belonging to different intents. We propose a simple algorithm to refine SIMVECS vectors based on the provided constraints: Whenever a "Must-Link" constraint is provided for a pair of points, we collapse their 2 vectors into one vector in the space. This is achieved by taking the max value of each entry of the corresponding indexes. Thus, the resulting vector is closer to neighbors of both points, shrinking the distances between neighbors of both utterances.Moreover, because each feature (dimension) in the space is a representative utterance, after the collapsing of two utterances, we perform PCA to come up with the new dimensions after the user is done with the labeling.

On the other hand, whenever a "Cannot-Link" constraint is provided, the similarity score between the two utterances is set to zero in the corresponding entry in SimMatrix. This increases the distance between the two points and transitively, between the neighbors of these two points.
To evaluate the effieciency of the resulting vectore, we vary the amount of user-labeled data points and estimate the corresponding clustering purity scores. As shown in Fig. 3, semi-supervised learning can significantly improve the performance of the clustering algorithm. We compare the performance of SIMVECS to BOW, Seq-Auto-encoder, and Fast-text. We use constrained K-means (COP-Kmeans) (Wagstaff et al., 2001) as the semi-supervised learner for SIMVECS as well as all baseline techniques. We see that SIMVECS achieves its maximum gain against other techniques when the proportion of labeled data is small. This is very useful in our problem as it reduces the labeling effort required from the user side to assist the clustering algorithm. The results show improvements over all three baselines, while the difference between the approaches shrinks with more labeled data points as expected. Also we notice that with few labeled data points (from 10% to 30%), Seq-Auto-encoder and Fast-text representations are performing better than BOW. However, with more labeled data ($\geq 50\%$) BOW starts to perform better. The reason is that with more labeled data points, the training examples (constraints) start to cover most of the expressions that can be used for a particular intent.

7 Supervised learning with conversational text

Supervised learning is critical to language understanding services in order to identify both intents and entities in new utterances coming in the stream. We compare the performance of intent classifiers when SIMVECS is used against BOW, Seq-auto-encoders, and Fast-text vector representations. For all corpora, a linear SVM classifier is trained per intent in one-vs-all fashion. For each intent, the utterances that belong to that intent represent the positive class, whereas all other utterances represent the negative class. We apply 5-fold cross validation and calculate the average F1-Score across all runs. Fig. 4 shows the improvement in F1-Score with both variants over baseline

Get Weather	
Is Warm	**Weather Update**
1 Will it get warmer, in Holy Cross Wilderness	4 What's the weather in Waretown, Lebanon
2 Will it be warmer, in five years, in Slemp, Kansas	5 What's the weather like in Saint Regis Falls, ND
3 Will it be warm, in Powersville, Guam, 23 hours from now	6 Tell me the weather forecast for Carmichaels, , Gambia, at one am

Figure 2: Example of clustering with different levels of granularity. Without user feedback, it is not clear whether the clustering algorithm should put all the utterances in the same cluster (Get Weather) or split them into two separate clusters (Is Worm & Weather Update).

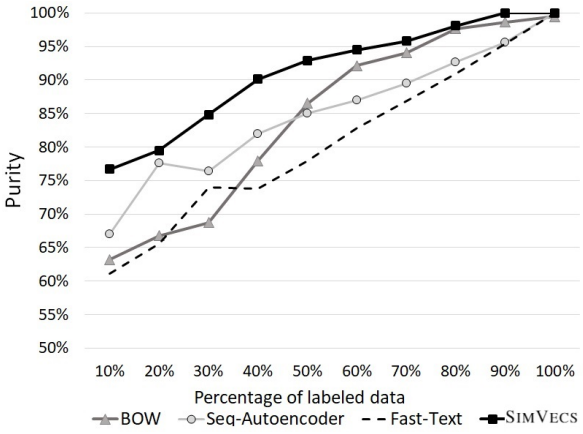


Figure 3: Improvement on clusters Purity using SIMVECS with COP-Kmeans vs several baselines

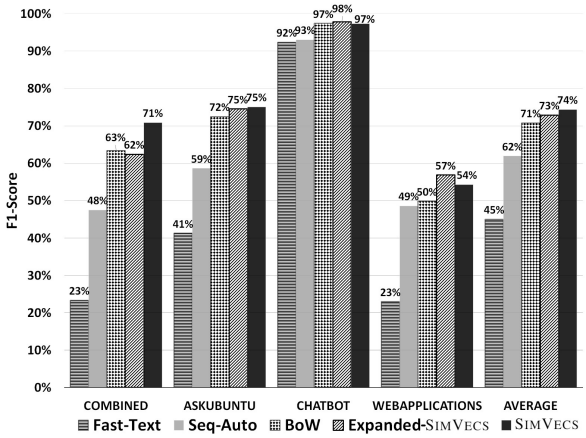


Figure 4: Improvement on classification accuracy using SIMVECS vs several baselines.

techniques. An improvement of 29% is observed over Fast-text representation, whereas an improvement of 13% over Seq-Auto-encoder representation is observed. SIMVECS is only slightly better than BOW (3% improvement on average). Additionally, we introduce a variant of SIMVECS that doesn't use the automatically tuned weights shown in Eq. 4. Instead, we concatenate all 6 similarity sub-metrics with all other utterances into one vector and use the resulting vector as the utterance representation (called Expanded-SIMVECS). Notice that this approach generates vector representations of size 6X compared to SIMVECS. We notice that using SIMVECS with our pre-trained weights is achieving better results than Expanded-SIMVECS across all corpora. The reason is that as Expanded-SIMVECS increases the number of dimensions, it also increases the sparsity of the space and hence requires more training data, which is known as "the curse of dimensionality" (Poggio et al., 2017). We also notice that the performance gain is proportional to the number of intents in the corpus. The peak gains of 8%, 25%, and 48% over BOW, Seq-Auto-encoder, and Fast-text respectively are observed with "Combined" corpora (which combines the 15 intents in all three corpora).

8 Background

In this section, we give the formulas used to estimate our evaluation metrics: Purity (Manning et al.) and F1-Score (Sasaki et al., 2007).

8.1 Purity:

is evaluated by the given formula:

$$Purity = \frac{1}{N}\sum_{i=1}^{k} Max_j |C_i \cap t_j| \quad (6)$$

Where N is the number of data points, k is the number of clusters, C_i is a generated cluster, and t_j is the intent which represents the majority in C_i.

8.2 F1-Score:

is evaluated by the given formula:

$$F_1 = 2.\frac{P * R}{P + R} \quad (7)$$

Where P is the precision, and R is the recall.

9 Discussion

One of the practical challenges in implementing SIMVECS can be the size of the SimMatrix, particularly when the number of utterances grows very large. Currently, this is not be a critical issue for current LU services as they tend to limit the number of utterances per application to a few thousands. But the maximium number of supported utterances is expected to grow in the future. For example, IBM Watson currently limits the number of utterances to 25,000 per workspace (application) whereas Microsoft LUIS limits the number of utterances to 15,000 per application. One approach to improve the scalability of SIMVECS is by constraining further the number of dimensions of the vector space thus reducing the memory requirements for storing SimMatrix. For reducing the computational time, multi-threading can be used to calculate similarity scores between different pairs of utterances concurrently and hence speedup the matrix construction process.

10 Conclusion

This paper introduces SIMVECS, a similarity-based vector representation technique designed to overcome prior work limitations in the field of conversational AI. We discussed the main challenges in vector representation for conversational AI applications and how SIMVECS overcomes these challenges. Through evaluation on different corpora and for different learning tasks, we showed the efficacy of vector representations generated by SIMVECS over several baselines.

References

Palakorn Achananuparp, Xiaohua Hu, and Xiajiong Shen. 2008a. The evaluation of sentence similarity measures. In *International Conference on data warehousing and knowledge discovery*, pages 305–316. Springer.

Palakorn Achananuparp, Xiaohua Hu, Xiaohua Zhou, and Xiaodan Zhang. 2008b. Utilizing sentence similarity and question type similarity to response to similar questions in knowledge-sharing community. In *Proceedings of QAWeb 2008 Workshop, Beijing, China*, volume 214.

Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text clustering algorithms. In *Mining text data*, pages 77–128. Springer.

Florian Beil, Martin Ester, and Xiaowei Xu. 2002. Frequent term-based text clustering. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 436–442. ACM.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Martin Boyanov, Ivan Koychev, Preslav Nakov, Alessandro Moschitti, and Giovanni Da San Martino. 2017. Building chatbots from forum data: Model selection using question answering metrics. *arXiv preprint arXiv:1710.00689*.

Daniel Braun, Adrian Hernandez-Mendez, Florian Matthes, and Manfred Langen. 2017. Evaluating natural language understanding services for conversational question answering systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 174–185.

Christian Buchta, Martin Kober, Ingo Feinerer, and Kurt Hornik. 2012. Spherical k-means clustering. *Journal of Statistical Software*, 50(10):1–22.

Mengen Chen, Xiaoming Jin, and Dou Shen. 2011. Short text classification improved by learning multi-granularity topics. In *IJCAI*, pages 1776–1781.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. 2013. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 45(3):35.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science*, 315(5814):972–976.

Frauke Friedrichs and Christian Igel. 2005. Evolutionary tuning of multiple svm parameters. *Neurocomputing*, 64:107–117.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Anna Huang. 2008. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

HK Lam, SH Ling, Frank HF Leung, and Peter Kwong-Shun Tam. 2001. Tuning of the structure and parameters of neural network using an improved genetic algorithm. In *Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE*, volume 1, pages 25–30. IEEE.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Yuhua Li, David McLean, Zuhair A Bandar, Keeley Crockett, et al. 2006. Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge & Data Engineering*, (8):1138–1150.

Ashraf Mahgoub, Paul Wood, Sachandhan Ganesh, Subrata Mitra, Wolfgang Gerlach, Travis Harrison, Folker Meyer, Ananth Grama, Saurabh Bagchi, and Somali Chaterji. 2017. Rafiki: a middleware for parameter tuning of nosql datastores for dynamic metagenomics workloads. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*, pages 28–40. ACM.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schutze. Introduction to information retrieval? cambridge university press 2008. *Ch*, 20:405–416.

Paul Mcnamee and James Mayfield. 2004. Character n-gram tokenization for european language text retrieval. *Information retrieval*, 7(1-2):73–97.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Melanie Mitchell. 1996. An introduction to genetic algorithms mit press. *Cambridge, Massachusetts. London, England*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. 2017. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing*, 14(5):503–519.

Yutaka Sasaki et al. 2007. The truth of the f-measure. *Teach Tutor mater*, 1(5):1–5.

Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. 2001. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Mohamed A Zahran, Ahmed Magooda, Ashraf Y Mahgoub, Hazem Raafat, Mohsen Rashwan, and Amir Atyia. 2015. Word representations in vector space and their applications for arabic. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 430–443. Springer.

Incorporating Interlocutor-Aware Context into Response Generation on Multi-Party Chatbots

Cao Liu[1,2], **Kang Liu**[1,2], **Shizhu He**[1,2], **Zaiqing Nie**[3], **Jun Zhao**[1,2]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Alibaba AI Labs, Beijing, 100029, China
{cao.liu, kliu, shizhu.he, jzhao}@nlpr.ia.ac.cn
zaiqing.nzq@alibaba-inc.com

Abstract

Conventional chatbots focus on two-party response generation, which simplifies the real dialogue scene. In this paper, we strive toward a novel task of Response Generation on Multi-Party Chatbot (RGMPC), where the generated responses heavily rely on the interlocutors' roles (e.g., speaker and addressee) and their utterances. Unfortunately, complex interactions among the interlocutors' roles make it challenging to precisely capture conversational contexts and interlocutors' information. Facing this challenge, we present a response generation model which incorporates Interlocutor-aware Contexts into Recurrent Encoder-Decoder frameworks (ICRED) for RGMPC. Specifically, we employ interactive representations to capture dialogue contexts for different interlocutors. Moreover, we leverage an addressee memory to enhance contextual interlocutor information for the target addressee. Finally, we construct a corpus for RGMPC based on an existing open-access dataset. Automatic and manual evaluations demonstrate that the ICRED remarkably outperforms strong baselines.

1 Introduction

Human computer conversation has been an important and challenging task in NLP and AI since the Turing Test was proposed in 1950 (Turing, 1950). Recently, with the rapid growth of social conversation data available on the Internet, data-driven chatbots are able to learn to generate responses directly and have attracted much more attention than before (Li et al., 2016a; Tian et al., 2017).

Researches in this area mostly focus on the dialog with two interlocutors (Maíra Gatti de Bayser et al., 2017). However, the real-life interaction involves a substantial part of Multi-Party Chatbots (MPC, such as internet forum and chat group), which is a form of conversation with multiple interlocutors (Ouchi and Tsuboi, 2016). For example, there are more than three interlocutors $(a_1, a_2, a_3 ... a_m)$ involved in the conversation in Figure 1, and their roles (e.g., speaker and addressee) may change across different dialog turns.

	t	Speaker	Addressee	Utterance
Context	1	Alan (a_1)	Bert (a_2)	the main ubuntu channels require that …
	2	Carl (a_3)	-	i found that worse like a 1 year ago …
	…	…	…	…
	n-1	Carl (a_3)	Jack (a_m)	i have a " dash to dock " extension …
	n	Alan (a_1)	Carl (a_3)	gnome classic is still available …
	n+1	Jack (a_m)	Carl (a_3)	mate is a recent reincarnation i …
		Responding Speaker	Target Addressee	(Generated) Response

Figure 1: An example of Multi-Party Chatbots (MPC). At each turn, a speaker said one utterance to an addressee. There are many interlocutors (e.g., Alan, Bert and so on) in a conversation, where a_i represents interlocutor's ID.

As shown in Figure 1, at each turn, the core issue of MPC is to capture *who* (speaker) talks to *whom* (addressee) about *what* (utterance). In order to obtain responses in MPC, in our best knowledge, previous approaches usually employ a response selection paradigm, which simply selects one response from a set of existing utterances as the final response according to the contexts. Obviously, this paradigm, which could not generate new responses, is not so flexible. In this study, to build a more broadly applicable system, we concentrate on producing new responses word by word, named as Response Generation on Multi-Party Chatbots (RGMPC).

RGMPC is a very challenging task. The primary challenge is that the generated response has strong relevance to the interlocutor's roles, such as the speaker and the addressee. For example, in the same context of Figure 1, what a_1 says to a_2 is different from what a_1 says to a_3 because different addressees (a_2 and a_3) have different information demands. Similarly, as for the same addressee, ut-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 718–727
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

terances from different speakers may be different because each speaker has personal background knowledge and style of speaking. Moreover, the roles of the same interlocutor may vary across different dialog turns. For instance, in Figure 1, a_3 plays different roles in different dialog turns: speaker in the turn 2 and n-1, addressee in the turn n and n+1.

Therefore, it is very important for RGMPC to capture interlocutor information. Currently, most response generation methods consider only the contextual utterance information (Serban et al., 2016, 2017) but neglect the interlocutor information. Although some researches have exploited the interlocutor information for response generation, they are still suffering from certain critical limitations. Li et al. (2016b) learn a fixed vector for each person from all conversational texts in the training corpus. However, as a global representation, the fixed person vector needs to be trained from large-scale dialogue turns for each interlocutor, and it may have a **sparsity issue** since some interlocutors have very few dialogue turns.

To address the aforementioned problems of RGMPC, this paper incorporates Interlocutor-aware Contexts into a Recurrent Encoder-Decoder model (ICRED) for RGMPC, which is also an end-to-end framework. Specifically, in order to capture interlocutor information, we exploit interactive interlocutor representations learned from current dialog context rather than the fixed person vectors (Li et al., 2016b) obtained from all dialogs in the training corpus. We expect that the learned contextual interlocutor representation could be a good alternative to the fixed person vectors (Li et al., 2016b) due to its ability of alleviating the sparsity issue. Furthermore, from the view of conversation analysis, responses are usually used for answering the addressee's question or expanding the addressee's utterances. Therefore, we originally introduce an addressee memory mechanism to enhance contextual information for the target addressee especially. Finally, both of the interactive interlocutor representation and addressee memory are utilized for decoding response utterances. In particular, the addressee memory is leveraged to capture the addressee information for each generated word dynamically.

In order to prove the effectiveness of the proposed model, we construct a dataset for RGMPC based on an open dataset[1]. Experimental results show that the proposed model is fairly competitive on both automatic and manual evaluations compared with state-of-the-arts.

In brief, the main contributions of the paper are as follows:

(1) We propose an end-to-end response generation model called ICRED which incorporates Interlocutor-aware Contexts into Recurrent Encoder-Decoder framework for RGMPC.

(2) We leverage an addressee memory mechanism to enhance contextual interlocutor information for the addressee.

(3) We construct an open-access dataset for RGMPC. Both automatic and manual evaluations demonstrate that our model is remarkably better than strong baselines in this dataset.

2 Task Formulation

	Data	Notation
Input	Context	$\mathcal{C} = [(a^t_{spk}, a^t_{adr}, \mathbf{u}^t)]^n_{t=1}$
	Responding Speaker	a^{n+1}_{spk} (or a_{res})
	Target Addressee	a^{n+1}_{adr} (or a_{tgt})
Output	Response	$\mathbf{u}^{n+1}$ (or $\{r_j\}^{L_r}_{j=1}$)

Table 1: Notations for RGMPC.

On multi-party chatbots, lots of interlocutors talk about one or more topics. At each dialogue turn (or time step) t, there is a speaker (a^t_{spk}), who may talk something ($\mathbf{u}^t$) to a specific addressee (a^t_{adr}), while the others are observers. As shown in Table 1, given the context $\mathcal{C}$ of previous n dialog turns, the responding speaker a_{res} and the target addressee a_{tgt} at time step n+1, the task of RGMPC aims to automatically generate the next utterance $\mathbf{u}^{n+1}$ as the final response. Here, $\mathcal{C}$ is a list ordered by the time step t: $\mathcal{C} = [\mathcal{C}^t]^n_{t=1} = [(a^t_{spk}, a^t_{adr}, \mathbf{u}^t)]^n_{t=1}$, where $\mathcal{C}^t$ means a^t_{spk} says $\mathbf{u}^t$ to a^t_{adr} at time step t, n is the maximum number of previous dialog turns in a context. $\mathbf{u}^t = (w^t_1, w^t_2 ... w^t_{L_u})$ is the input utterance (word sequence) at time step t, where L_u is the number of maximum words in utterances.

3 Methodology

The overview of the proposed ICRED for RGMPC is shown in Figure 2 along with its caption. The details are as follows.

[1]The dataset is available at https://www.dropbox.com/s/4chh64yaxajh0j7/RGMPC.zip?dl=0

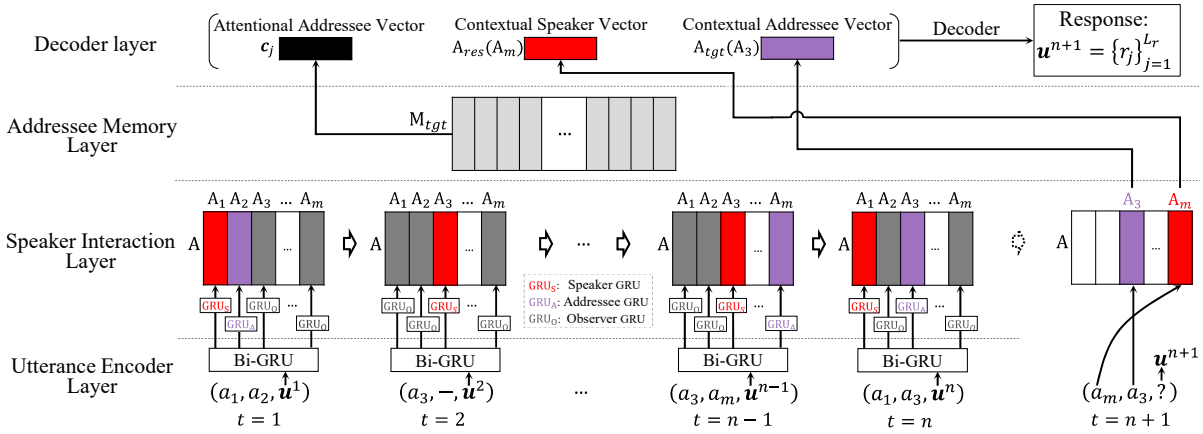


Figure 2: Overall structure of the proposed ICRED for RGMPC. At each time step t, $(a_i, a_j, \mathbf{u}^t)$ means that a speaker a_i said an utterance $\mathbf{u}^t$ to an addressee a_j, where the time step t is denoted on the bottom, and the superscript t may be omitted for brevity. Our ICRED includes: ① Utterance Encoder Layer: encoding each utterance ($\mathbf{u}^t$) into distributed vectors; ② Speaker Interaction Layer: capturing interactive interlocutor information from contexts, and it updates all interlocutors' representation by different GRUs according to their roles at each time step, where the embedding for an interlocutor a_i is obtained by extracting the i-th column (A_i) from the interlocutor embedding matrix A; ③ Addressee Memory Layer: enhancing contextual information for the target addressee (a_3); ④ Decoder Layer: generating responses.

3.1 Utterance Encoder Layer

The utterance encoder layer transforms input utterance into distributional representations. We leverage the bi-directional Gated Recurrent Units (GRU) (Cho et al., 2014) to capture the long-term dependency. For an utterance $\mathbf{u}^t = (w_1^t, w_2^t ... w_{L_u}^t)$ at time step t, the concatenated representation for hidden states in bi-directions is denoted as $\mathbf{h}_i^t = [\overrightarrow{\mathbf{h}}_i^t, \overleftarrow{\mathbf{h}}_{L_u-i+1}^t]$, where $\mathbf{h}_i^t$ is considered as the contextual word representation of the input word w_i^t. The state ($\mathbf{h}_{L_u}^t$) of the last word is treated as the representation of the utterance at time step t, which is denoted as $\mathbf{h}^t$, and it could be sent to the speaker interaction layer for updating contextual representation.

3.2 Speaker Interaction Layer

The speaker interaction layer is leveraged to obtain the interlocutor information in the context. Similar to the Speaker Interaction RNNs (Zhang et al., 2018), we utilize the interactive speaker encoder for RGMPC.

As shown in Figure 2, an interlocutor embedding matrix A is used to record all interlocutors' representation, and A is initiated with a zero matrix. Each column of A corresponds to an interlocutor's embedding: $A_i = A[*, a_i]$, where A_i is the embedding for the interlocutor a_i. The speaker interaction layer updates the entire interlocutors' embeddings at each time step based on their roles (speaker, addressee or observer). Embeddings for the speaker, addressee and observer are updated by following role-differentiated GRUs: GRU_S, GRU_A and GRU_O, respectively.

$$A_{spk}^t = \mathrm{GRU}_S(A_{spk}^{t-1}, \mathbf{h}^t) \quad (1)$$

$$A_{adr}^t = \mathrm{GRU}_A(A_{adr}^{t-1}, \mathbf{h}^t) \quad (2)$$

$$A_{obv}^t = \mathrm{GRU}_O(A_{obv}^{t-1}, \mathbf{h}^t) \quad (3)$$

where A_{spk}^t (A_{adr}^t / A_{obv}^t) is the embedding for the speaker (addressee / observer) at time step t, and $\mathbf{h}^t$ is the utterance representation obtained from the utterance encoder layer. Take the first time step "$(a_1, a_2, \mathbf{u}^1)$" in Figure 2 as an example, when a_1 says $\mathbf{u}^1$ to a_2, the speaker's (a_1's) embedding A_1 is updated by the speaker GRU—GRU_S, and the addressee's (a_2's) embedding A_2 is updated by the addressee GRU—GRU_A, while other interlocutors' embeddings are updated by the observer GRU—GRU_O. Note that the addressee may be missing (such as "$(a_3, -, \mathbf{u}^2)$" at time step 2 in Figure 2), where embeddings for all interlocutors except for the speaker are updated by the observer GRU. The interlocutor embedding matrix (A) is updated up to the maximum time step n. The final interlocutor embedding matrix is used in decoding.

3.3 Addressee Memory Layer

The interlocutor embedding matrix is updated by utterance representations and interlocutor's roles,

so it captures interlocutor's context on the utterance level. In fact, contextual word representation is important for response generation, too. A context contains consecutive utterances, and each utterance is a word sequence. Therefore, memorizing all contextual word representations in the entire context is complex, and it is difficult to work on large-scale utterances in one context.

Intuitively, from the view of conversational analysis, responses are usually used for answering the addressee's question or expanding the addressee's utterances. Therefore, we design an addressee memory layer, which only memorizes the contextual word representations (noted as M_{tgt}) in the last utterance said by the target addressee, and the contextual representation for each word is obtained from the utterance encoder layer. Take "$(a_m, a_3, ?)$" at time step n+1 in Figure 2 as an example, $\mathbf{u}^{n-1}$ is the last utterance said by the target addressee a_3 because of "$(a_3, a_m, \mathbf{u}^{n-1})$" at time step n-1, so the addressee memory layer merely memorizes contextual word representation $M_{tgt} = [\mathbf{h}_1^{n-1}, \mathbf{h}_2^{n-1}, ..., \mathbf{h}_{L_u}^{n-1}]$ from the utterance $\mathbf{u}^{n-1}$, where $\mathbf{h}_i^{n-1}$ is obtained from Section 3.1.

3.4 Decoder Layer

The decoder is responsible for generating target sequences. Different from a single contextual representation in previous work (Serban et al., 2017), the speaker interaction layer is able to capture different interlocutor information from contexts (e.g., personal background knowledge and style of speaking for the responding speaker, special information demands for the target addressee). Moreover, the addressee memory layer records contextual word representation for the target addressee. Therefore, we extract **contextual speaker vector** A_{res} for the responding speaker a_{res} from the final interlocutor embedding matrix A (e.g., the responding speaker's embedding obtained by $A_m = A[*, a_m]$ for the responding speaker a_m in Figure 2). Similarly, **contextual addressee vector** A_{tgt} for the target addressee is also extracted from A. However, A_{res} and A_{tgt} keep same for each generated word. In order to capture dynamic information for different generated words, we leverage an attention mechanism to selectively reads different contextual word representations from the addressee memory. For each target word, the decoder attentively reads the contextual word representation as follows:

$$\mathbf{c}_j = \sum_{k=1}^{L_u} \alpha_{jk} M_{tgt}[*, k]; \tag{4}$$

$$\alpha_{jk} = \frac{e^{\rho(\mathbf{s}_{j-1}, M_{tgt}[*,k])}}{\sum_{k'} e^{\rho(\mathbf{s}_{j-1}, M_{tgt}[*,k'])}} \tag{5}$$

where $\mathbf{c}_j$ is the **attentional addressee vector**, $M_{tgt}[*, k]$ is the contextual word representation for the k-th word in the addressee memory, and $\mathbf{s}_j$ represents the hidden state in decoding GRU. A function ρ is leveraged to compute the attentive strength, which is calculated by a projected matrix to connect $\mathbf{s}_{j-1}^T$ and $M_{tgt}[*, k]$. Finally, the attentional addressee vector $\mathbf{c}_j$, contextual speaker vector A_{res} and contextual addressee vector A_{tgt} are concatenated to estimate the probability for predicted words:

$$p(r_j | r_{<j}, a_{spk}, a_{tgt}, \mathcal{C}) = p(r_j | r_{j-1}, \mathbf{c}_j, A_{res}, A_{tgt}, \mathbf{s}_j) \tag{6}$$

$$\mathbf{s}_j = \mathrm{GRU}_{dec}(\mathbf{s}_{j-1}, [\mathbf{c}_j, A_{res}, A_{tgt}, \mathbf{x}_{j-1}]) \tag{7}$$

where $\mathbf{s}_j$ is the hidden state of the decoding GRU—$\mathrm{GRU}_{\mathrm{dec}}$. $\mathbf{x}_j$ is the word vector of the predicted target word r_j, and r_j is typically performed by a $softmax$ classifier over a settled vocabulary based on word embedding similarity.

3.5 Learning

The proposed ICRED for RGMPC is totally differentiable, and it can be optimized in an end-to-end manner using back-propagation. Given the context $\mathcal{C}$, responding speaker a_{res}, target addressee a_{tgt} and target word sequence $\{r_j\}_{j=1}^{L_r}$, the objective function is to minimize the loss function:

$$\mathcal{L} = \frac{-1}{L_r} \sum_{j=1}^{L_r} log[p(r_j | r_{<j}, \mathcal{C}, a_{res}, a_{tgt}] + \lambda \mathcal{L}_2 \tag{8}$$

It contains a negative log-likelihood for generated responses and L2 regularization ($\mathcal{L}_2$), where λ is a hyperparameter for $\mathcal{L}_2$.

4 Experiment

4.1 Dataset

Our dataset is constructed based on the Ubuntu multi-party chatbot corpus[2], which has been widely used as the evaluation dataset for the response selection task (Ouchi and Tsuboi, 2016; Zhang

[2]https://github.com/hiroki13/response-ranking

	Total	Train	Dev	Test
# Contexts	423.5K	338.9K	42.3K	42.3K
# Speaker	35.3K	33.5K	15.6K	15.6K
# Addressee	23.4K	22.4K	10.8K	10.8K
# Vocab	276.1K	254.8K	82.2K	82.0K
# Tokens	26.3M	21.0M	2.62M	2.62M
Avg. Tok/Ctx	51.4	51.5	51.4	51.3
Avg. Tok/Res	10.6	10.6	10.7	10.6

Table 2: Data statistics. "#" means number, and "Avg. Tok/Ctx (or Res)" is the number of tokens per context (or response).

et al., 2018). The original data comes from the Ubuntu IRC chat log, where each line consists of (Time, Speaker, Utterance). If the addressee is explicitly mentioned in the utterance, it is extracted as the addressee. Otherwise, all interlocutors except the speaker are observers. Considering that generating new responses in this paper is more complicated than retrieving responses, the generative task requires higher-quality data. We suppose that the responding speaker and target addressee have appeared in the context, where the contextual window is set to 5. Moreover, the words are tokenized by NLTK, and some general responses are removed by human rules[3]. Finally, we randomly split the dataset into Train/Dev/Test (8:1:1), and it is publicly available[1]. The detailed statistics of the dataset are shown in Table 2.

4.2 Implement Details

In order to keep our model comparable to other typical existing methods, we keep the same parameters and experimental environments for ICRED and the comparative models. We take a maximum of 20 words for the utterance. The word vector dimension is 300 and it is initialized with the public released fasttext[4] pre-trained on Wikipedia. The utterance and interlocutor are encoded by 512-dimensional and 1024-dimensional vectors, respectively. The joint loss function with 0.0001 L2 weight is minimized by an Adam optimizer. We implemented all the models with Tensorflow on an NVIDIA TITAN X GPU.

4.3 Automatic Evaluation Metrics

Automatic evaluations (AEs) for Natural Language Generation (NLG) is a challenging and under-researched problem (Novikova et al., 2017). Following (Liu et al., 2018), we leverage two referenced measurements (BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004)[5]) for automatic evaluations. Considering that current data-driven approaches tend to generate short and generic (meaningless) responses, two unreferenced ("intrinsic") metrics are also leveraged to the evaluation. The first one is the average length of responses, which is an objective and surfaced metric reflected the substance of responses (Mou et al., 2016; He et al., 2017a). The other one is the number of nouns[6] per response (Liu et al., 2018), which shows the richness of responses since nouns are usually content words. Note that the unreferenced metrics could enrich the evaluations, though they are weak metrics. The detailed results and analyses are shown as follows.

4.4 The Effectiveness of ICRED for RGMPC

Model	Referenced		Unreferenced	
	BLEU	ROUGE	Length	#Noun
Seq2Seq	8.86	7.62	9.48	1.24
Persona Model	9.12	7.38	11.04	1.29
VHRED	9.38	7.65	10.25	1.55
ICRED (ours)	**10.63**	**8.73**	**11.34**	**1.68**

Table 3: Overall comparisons of ICRED.

Comparison Methods. We compared ICRED with the following methods:

(1) Seq2Seq (Sutskever et al., 2014): Seq2Seq is one of the mainstream methods for text generation. In order to capture as much information as possible, the input sequence is all utterances concatenated in order in a context.

(2) Persona Model (Li et al., 2016b): The persona-based model modified a Seq2Seq to encode a global vector for each interlocutor that appears in the training data, and it could alleviate the issue of speaker consistency for response generation.

(3) VHRED (Serban et al., 2017): VHRED is essentially a conditional variational auto-encoder with hierarchical encoders, and it extends HRED (Serban et al., 2016) by adding a high-dimensional latent variable for utterances.

Comparative Results. Table 3 demonstrates overall comparisons of ICRED. We can clearly obtain the following observations:

[3]We list some general responses, such as containing "*i don't know*", "*you are welcome*".

[4]https://github.com/facebookresearch/fastText

[5]Implemented by https://github.com/Maluuba-/nlg-eval. BLEU and ROUGE are transformed into percentages (%).

[6]NLTK is utilized for part-of-speech tagging.

Interlocutor's Dialogue Turns	Persona Model		ICRED (ours)	
	BLEU	ROUGE	BLEU	ROUGE
[0, 100]	8.47	6.72	10.63	8.60
(100, 1000]	8.87	7.14	10.50	8.61
(1000, 5000]	9.48	7.74	**10.77**	**8.90**
(5000, +∞)	**9.51**	**7.80**	10.60	8.79

Table 4: Performances on sparse and plentiful learning data with different numbers of interlocutor's dialogue turns, where the test data is divided into different intervals according to the number of dialogue turns in training dataset said by target addressee (named as interlocutor's dialogue turns).

(1) ICRED obtains the highest performance on all metrics (marked as **bold**), and it indicates that incorporating interlocutor-aware context into RGMPC contributes to generating better responses.

(2) Although the persona-based model utilizes interlocutor information, it performs poorly. The average dialogue turn for the interlocutor is more than 5000 in (Li et al., 2016a), while there is less than 100 dialogue turns per interlocutor in our dataset. Therefore, it is hard to learn a global vector for each interlocutor from the sparse corpus. In contrast, our ICRED performs well on such a sparse corpus (details in Section 4.5).

(3) VHRED brings slight improvements over the Seq2Seq and persona-base model. Even that VHRED enhances the contextual information by a high-dimensional latent variable, VHRED is still remarkably worse than ICRED because VHRED neglects the interlocutor information.

4.5 The Effect of Sparse Data on ICRED

Comparison Settings. Persona model (Li et al., 2016b) may have a sparsity issue since some interlocutors have very few dialogue turns. To investigate whether ICRED has the sparsity issue or not, we divide the test data into four intervals according to the number of training dialogue turns said by the target addressee (called interlocutor dialogue turns), where small turns represent sparse learning data (e.g., "[0, 100]") and large turns mean plentiful learning data (e.g., "(5000, +∞)").

Comparative Results. Table 4 reports the performances of persona model and ICRED on different interlocutor's dialogue turns for learning. We can clearly see that the persona model has a sparsity issue: it performs very poorly on sparse learning data (e.g., BLEU score = 8.47 on "[0, 100]") while it achieves good performances on plentiful learning data (e.g., BLEU score = 9.51 on "(5000, +∞)"), which demonstrates that the fixed person vectors in the persona model need to be learned from large-scale training data for each interlocutor. In contrast, ICRED exploits interactive interlocutor representation learned from current dialog context rather than the fixed person vectors obtained from all training dialog utterances. Therefore, ICRED has no sparsity issues and it performs closely on sparse and plentiful learning data.

4.6 Ablation Study for Model Components

Model	Referenced		Unreferenced	
	BLEU	ROUGE	Length	#Noun
ICRED	**10.63**	**8.73**	**11.34**	**1.68**
w/o Adr_Mem	10.25	8.23	10.73	1.27
w/o Ctx_Spk_Vec	10.13	8.22	10.86	1.59
w/o Ctx_Adr_Vec	9.95	8.18	10.93	1.26

Table 5: Ablation Experiments by removing the main components.

Comparison Settings. In order to validate the effectiveness of model components, we have tried to remove some main components in decoding as follows. (1) w/o Adr_Mem: without the addressee memory, such as removing $\mathbf{c}_j$ in Equation 6-7; (2) w/o Ctx_Spk_Vec: without the contextual speaker vector, such as removing A_{res} in Equation 6-7; (3) w/o Ctx_Adr_Vec: without the contextual addressee vector, such as removing A_{tgt} in Equation 6-7.

Comparative Results. Results of the ablation study are shown in Table 5. We can see that removing any component causes obvious performance degradation. In particular, "w/o Ctx_Adr_Vec" performs the worst on almost all of the metrics, which demonstrates the importance of contextual information for the target addressee.

4.7 The Effectiveness of Addressee Memory

Memory Type	Referenced		Unreferenced	
	BLEU	ROUGE	Length	#Noun
addressee memory	**10.63**	8.73	11.34	**1.68**
all utterance memory	10.39	**8.78**	**11.38**	1.37
latest memory	10.43	8.40	10.16	1.28
speaker memory	10.03	8.28	10.72	1.66
w/o memory	10.25	8.23	10.73	1.27

Table 6: Performances over different memory types.

Comparison Settings. In order to demonstrate the effectiveness of the addressee memory, we

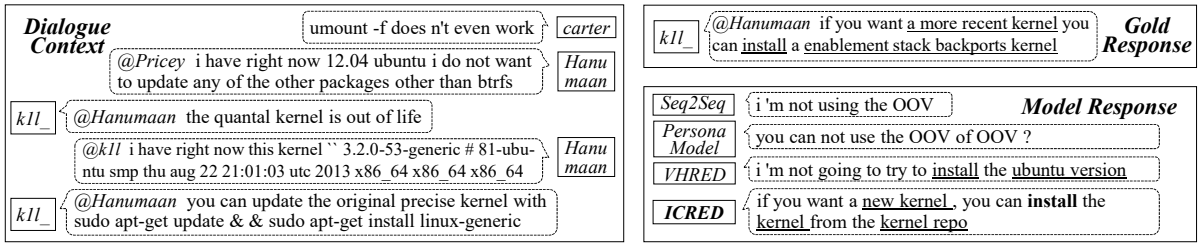


Figure 3: An example of different model responses for the same dialogue context. The input dialogue context is on the left. The gold (referenced) response and model responses are on the top right and bottom right, respectively. The rounded rectangle is the message box, where the *italic* behind "@" is the addressee, and the solid-line box near to the message box represents the speaker or model.

change the memory type, and then the attention model in Equation 5 is based on the new memory. The comparison settings are shown as follows. (1) addressee memory: memorizing contextual word representations in the last utterance said by the *target addressee* (e.g., $\mathbf{u}^{n-1}$ in Figure 2); (2) all utterance memory: memorizing contextual word representations in *all* utterances of the context (e.g., $\mathbf{u}^1$ to $\mathbf{u}^n$ in Figure 2); (3) latest memory: memorizing contextual word representations of the *latest* utterance in the context (e.g., the latest utterance $\mathbf{u}^n$ in Figure 2); (4) speaker memory: memorizing contextual word representations in the last utterance said by the *responding speaker*; (5) w/o memory: without any memory.

Comparative Results. We report the results of different memory types as shown in Table 6. It can see that our method, the addressee memory, achieves the best or near-best performances on all metrics. Although memorizing all utterances is competitive, the complexity of all utterance memory is n times compared with the one in the addressee memory, where n is the number of utterances in a context. The speaker memory performs closely to without memory, which indicates that not all memories can improve the performance.

4.8 Manual Evaluations

Besides automatic evaluations, we employ manual evaluations (MEs), which is important for response generation. Similar to (He et al., 2017b; Zhou et al., 2018), and we select three metrics for MEs, which measure the following aspects. (1) Fluency: measuring whether responses are grammatically correct or wrong. (2) Consistency: measuring whether responses are coherent to the context or not. (3) Informativeness: measuring how much informational (knowledgeable) content obtained from the responses.

Model	Flu.	Con.	Inf.
ICRED vs. Seq2Seq	77.25	83.69	84.35
ICRED vs. Persona.	78.44	80.41	82.35
ICRED vs. VHRED	73.20	81.29	79.47

Table 7: Manual evaluations (%) with fluency (Flu.), consistency (Con.) and informativeness (Inf.). The Score is the percentage that ICRED wins baselines after removing the "tie" pairs.

We conduct a pair-wise comparison between the response generated by ICRED and the one for the same input by three typical baselines. We sample 100 responses from each compared methods. Two curators judge (win, tie and lose) between these two methods. The Cohen Kappa of inter-annotator statistics is 0.750, 0.658 and 0.580 for the fluency, consistency and informativeness, respectively. As shown in Table 7, the score is the percentage that ICRED wins baselines after removing the "tie" pairs, and we can obtain that ICRED is significantly (sign test, p-value < 0.005) superior to all baselines on any metric. It demonstrates our model is able to deliver more fluent, consistent and informative responses.

4.9 Case Study

Figure 3 shows an example of responses on different models for the same dialogue context. It is clearly observed that our model (**ICRED**) generates more fluent, consistent and knowledgeable (marked as underline) responses compared to baselines. In particular, the response given by ICRED "*if you want a new kernel , you can install the kernel from the kernel repo*", not only explains the *reason for kernel installation* but also suggests a *source of the installation*. It fully captures the context and then produces a fluent, consistent and knowledgeable response, which is semantically similar to the gold one.

4.10 Discussion

Interlocutor Prediction and RGMPC. The above methods assume that the responding speaker and target addressee are given for RGMPC. Though the speaker and the addressee could be obtained in some situations (e.g., extracted from chat logs), it is still a researchable task to interlocutor prediction. There have been some researches to predict either the responding speaker or the target addressee based on the given textual contexts or multimodal information (Akhtiamov et al., 2017a; Meng et al., 2017; Akhtiamov et al., 2017b). Nevertheless, in order to obtain the interaction between interlocutor prediction and RGMPC, we further design a joint model for RGMPC and interlocutor prediction. Note that both the speaker and the addressee are predicted based on textual contexts, simultaneously. Firstly, the responding speaker is predicted from contexts:

$$p(a_{res}|\mathcal{C}) = \sigma([\mathbf{h}_{\mathcal{C}}; \mathbf{h}^n_{L_u}] \cdot \mathbf{W} \cdot A_{res}) \quad (9)$$

where $\mathbf{h}_{\mathcal{C}}$ is a summary contextual vector, which is max-pooled by the final interlocutor embedding matrix (A), and $\mathbf{h}^n_{L_u}$ is the hidden state of the last utterance. $\mathbf{W}$ is a projected matrix. a_{res} and A_{res} are the ID and the embedding of the responding speaker, respectively. The responding speaker is predicted by a *softmax* classifier based on the embedding similarity, and the target addressee is obtained in the same way. Secondly, the predicted interlocutors replace the gold ones for the addressee memory and extracting interlocutor's embeddings from A. Finally, the interlocutor prediction loss is added to the response generation loss for training. Table 8 shows the response generation performance on the situation that responding interlocutors are given and predicted. We can observe that:

(1) The overall performance on predicted interlocutors ("* / *" in Table 8) is slightly worse than the one with gold interlocutors (the first line in Table 8). Nevertheless, "* / *" still outperforms the strongest baseline (VHRED in Table 3).

(2) The correctness of interlocutor prediction has a significant impact on response generation performance. It performs the best when the responding speaker and the target addressee are predicted correctly. "False / False" (both are mispredicted) obtains the worst performance on the referenced metrics. These results demonstrate that both responding speaker and target addressee contribute to generating better responses.

Person	Speaker / Addressee	Referenced		Unreferenced	
		BLEU	ROUGE	Length	#Noun
Gold	True / True	10.63	8.73	11.34	1.68
Predict	$*$ / $*$	9.62	7.88	11.99	1.44
	True / True	**10.05**	**8.36**	12.04	1.43
	True / $*$	9.91	8.18	11.95	1.43
	$*$ / True	9.89	8.21	11.97	1.43
	False / False	9.20	7.41	**12.18**	**1.47**

Table 8: Performance on learning interlocutor prediction and RGMPC. "True" and "False" means right and wrong interlocutor, respectively. "*" represents both "True" and "False". The correctness of the responding speaker and target addressee is segmented by "/". For example, "True / *" means that the responding speaker is right, and the target addressee is right or wrong.

(3) Surprisingly, the unreferenced metrics perform well on "False / False". One possible reason is that the wrong interlocutors also capture rich contexts, and it generates long and meaningful responses but with a weak correlation to the gold interlocutors. Therefore, it achieves very poor performance on the referenced metrics.

5 Related Work

Our work is inspired by a large number of applications utilizing recurrent encoder-decoder frameworks (Cho et al., 2014) on NLP tasks such as machine translation (Bahdanau et al., 2015) and text summarization (Chopra et al., 2016). Recently, many researches extend the encoder-decoder framework on response generation. HRED (Serban et al., 2016) utilizes hierarchical encoder to capture the context. VHRED (Serban et al., 2017) extends HRED by adding a high-dimensional latent variable for utterances. These researches demonstrate the importance of contexts on response generation.

Our work is also inspired by researches on multi-party chatbots. Dielmann and Renals (2008) automatically recognize dialogue acts in multi-party speech conversations. Recently, some studies focus on the three elements (speaker, addressee, response) on multi-party chatbots. Meng et al. (2017) introduce speaker classification as a surrogate task. Addressee selection is researched by (Akhtiamov et al., 2017b). Some researches strive to the response selection (Ouchi and Tsuboi, 2016; Zhang et al., 2018). However, the response selection heavily relies on the candidates, and it can not generate new responses in new dialogue contexts. Response generation could solve

this problem. Li et al. (2016b) learn fixed person vector for response generation. Unfortunately, it needs to be obtained from large-scale dialogue turns, which has a sparsity issue: some interlocutors have very little dialog data. Differently, our model has no such restrictions.

6 Conclusion

In this study, we formalize a novel task of Response Generation for Multi-Party Chatbots (RGMPC) and propose an end-to-end model which incorporates Interlocutor-aware Contexts into Recurrent Encoder-Decoder frameworks (ICRED) for RGMPC. Specifically, we employ interactive speaker models to capture contextual interlocutor information. Moreover, we leverage an addressee memory mechanism to enrich contextual information. Furthermore, we propose to predict both the speaker and the addressee when generating responses. Finally, we construct a corpus for RGMPC. Experimental results demonstrate the ICRED remarkably outperforms strong baselines on automatic and manual evaluation metrics.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No.61533018), the Natural Key R&D Program of China (No.2017YFB1002101), the National Natural Science Foundation of China (No.61702512) and the independent research project of National Laboratory of Pattern Recognition. This work was also supported by CCF-DiDi BigData Joint Lab.

References

Oleg Akhtiamov, Maxim Sidorov, Alexey A. Karpov, and Wolfgang Minker. 2017a. Speech and text analysis for multimodal addressee detection in human-human-computer interaction. In *Proceedings of INTERSPEECH*, pages 2521–2525.

Oleg Akhtiamov, Dmitrii Ubskii, Evgeniia Feldina, Alexey Pugachev, Alexey Karpov, and Wolfgang Minker. 2017b. Are you addressing me? multimodal addressee detection in human-human-computer conversations.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of ICLR*.

Paulo Rodrigo Cavalin Maíra Gatti de Bayser, Renan Souza, Alan Braz, Heloisa Candello, Claudio S. Pinhanez, and Jean-Pierre Briot. 2017. A hybrid architecture for multi-party conversational systems. *CoRR*, abs/1705.01214.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL*, pages 93–98.

Alfred Dielmann and Steve Renals. 2008. Recognition of dialogue acts in multiparty meetings using a switching dbn. *IEEE transactions on audio, speech, and language processing*, pages 1303–1314.

He He, Anusha Balakrishnan, Mihail Eric, and Percy Liang. 2017a. Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of ACL*, pages 1766–1776.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017b. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of ACL*, pages 199–208.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of NAACL*, pages 110–119.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016b. A persona-based neural conversation model. In *Proceedings of ACL*, pages 994–1003.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of ACL workshop*, page 10.

Cao Liu, Shizhu He, Kang Liu, and Jun Zhao. 2018. Curriculum learning for natural answer generation. In *Proceedings of IJCAI*, pages 4223–4229.

Zhao Meng, Lili Mou, and Zhi Jin. 2017. Hierarchical rnn with static sentence-level attention for text-based speaker change detection. In *Proceedings of CIKM*, pages 2203–2206.

Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2016. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING*, pages 3349–3358.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for nlg. In *Proceedings of EMNLP*, pages 2241–2252.

Hiroki Ouchi and Yuta Tsuboi. 2016. Addressee and response selection for multi-party conversation. In *Proceedings of EMNLP*, pages 2133–2143.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of AAAI*, pages 3776–3783.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of AAAI*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. 2017. How to make context more useful? an empirical study on context-aware neural conversational models. In *Proceedings of ACL*, pages 231–236.

Turing. 1950. Computing machinery and intelligence. *Mind*, pages 433–460.

Rui Zhang, Honglak Lee, Lazaros Polymenakos, and Dragomir Radev. 2018. Addressee and response selection in multi-party conversations with speaker interaction rnns. In *Proceedings of AAAI*.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In *Proceedings of IJCAI*, pages 4623–4629.

Memory Graph Networks for Explainable Memory-grounded Question Answering

Seungwhan Moon, Pararth Shah, Anuj Kumar, Rajen Subba
Facebook Assistant
{shanemoon, pararths, anujk, rasubba@}fb.com

Abstract

We introduce Episodic Memory QA, the task of answering personal user questions grounded on memory graph (MG), where episodic memories and related entity nodes are connected via relational edges. We create a new benchmark dataset first by generating synthetic memory graphs with simulated attributes, and by composing 100K QA pairs for the generated MG with bootstrapped scripts. To address the unique challenges for the proposed task, we propose Memory Graph Networks (MGN), a novel extension of memory networks to enable dynamic expansion of memory slots through graph traversals, thus able to answer queries in which contexts from multiple linked episodes and external knowledge are required. We then propose the Episodic Memory QA Net with multiple module networks to effectively handle various question types. Empirical results show improvement over the QA baselines in top-k answer prediction accuracy in the proposed task. The proposed model also generates a graph walk path and attention vectors for each predicted answer, providing a natural way to explain its QA reasoning.

1 Introduction

The task of question and answering (QA) has been extensively studied, where many of the existing applications and datasets have been focused on the fact retrieval task from a large-scale knowledge graph (KG) (Bordes et al., 2015), or machine reading comprehension (MRC) approaches given unstructured text (Rajpurkar et al., 2018). In this work, we introduce the new task and dataset for **Episodic Memory QA**, in which the model answers personal and retrospective questions based on **memory graphs** (MG), where each episodic memory and its related entities (*e.g.* knowledge graph (KG) entities, participants, ...) are represented as the nodes connected via corresponding edges (Figure 1). Examples of such queries include "*Where did we go **after** we had brunch with Jon?*", "*How many times did I go to **jazz** concerts last year?*", etc. For Episodic Memory QA, a machine has to understand the contexts of a question and navigate multiple MG episode nodes as well as KG nodes to gather comprehensive information to match the query requirement.

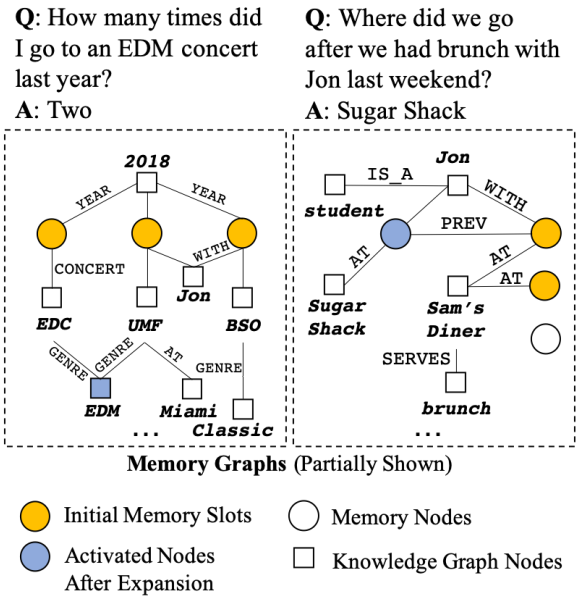


Figure 1: Illustration of **Episodic Memory QA** with user queries and memory graphs (MG) with knowledge graph (KG) entities. Relevant memory nodes are provided as initial memory slots via graph search lookup. The Memory Graph Network walks from the initial nodes to attend to relevant contexts and expands the memory slots when necessary. The main QA model takes these graph traversal paths and expanded memory slots as input, and predicts correct answers via multiple module networks (*e.g.* `COUNT`, `CHOOSE`, etc.)

While the ability of querying a personal database could lead to many potential applications, previous work in this domain (Jiang et al., 2018) is limited due to the lack of a large-scale

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 728–736
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

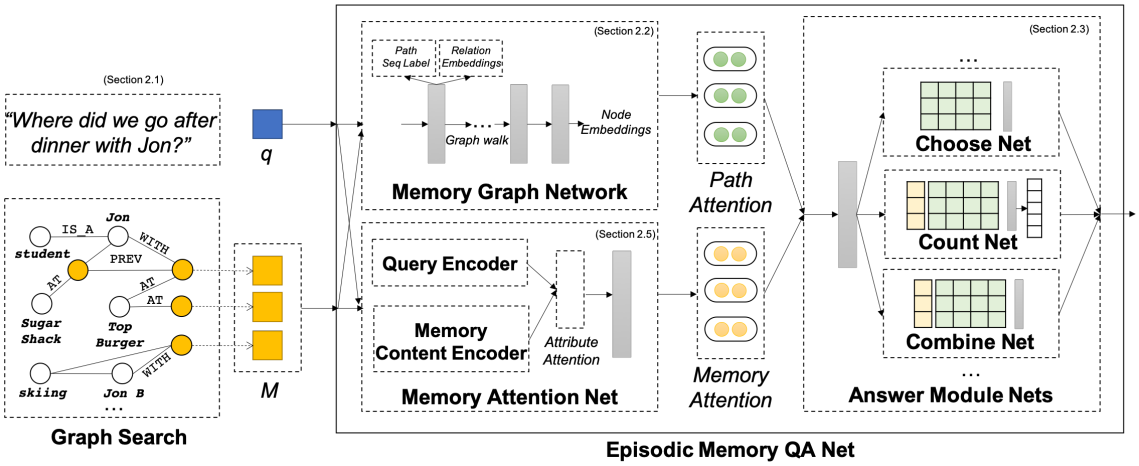


Figure 2: **Overall architecture** of the Episodic Memory QA Network. For an input query $\mathbf{q}$, candidate memory nodes $\mathbf{m} = \{\mathbf{m}^{(k)}\}$ are provided as input memory slots for the Memory QA Network. The Memory Graph Network then traverses the memory graph to expand the initial memory slots and activate other relevant entity and memory nodes based on the input queries. The Answer Module Networks execute the predicted neural programs to decode answers given the memory graph network outputs.

dataset and a unique set of challenges unseen in other tasks: For example, we observe that 1) Memory QA queries often include ambiguous and incomplete descriptions of reference memory (as opposed to many conventional Fact QAs with unambiguous mentions, *e.g.* "*Who painted the Mona Lisa?*"), hence requiring extensive candidate memory generation. Another challenge we observe is the case where 2) target memory is only indirectly linked to reference memory or entities (*e.g.* "*Where did we go after brunch?*"), which makes the conventional information retrieval (IR) approaches for generating answer candidates ineffective. In addition, 3) queries are not confined to retrieval tasks, but include various types of questions such as counting, set comparing, etc., many of which remain unsolved or not considered in many QA tasks.

To this end, we propose a new model called the Memory Graph Network (MGN) to address the specific challenges stated above that come with Memory QA. While Memory Networks have successfully been used in QA applications, typical limitations are that memory slots are limited to the fixed number of slots, often in sentence or bag-of-symbols forms. MGN extends the popular memory networks by storing graph nodes as memory slots and by allowing the network to dynamically expand memory slots through graph traversals. We then implement the main Episodic Memory QA Network with multiple module networks such as `CHOOSE`, `COUNT`, etc., to effectively handle various question types not easily handled via graph networks.

To bootstrap a large-scale dataset collection for Episodic Memory QA, we first build a synthetic memory graph generator, which creates multiple episodic memory graph nodes connected with real entities (*e.g.* locations, events, public entities) appearing on common-fact KGs. By creating a realistic memory graph that is synthetically generated, we avoid the need for inferring memory graphs from other structured data (*e.g.* photo albums) which are often limited in size. We then generate 100K QA pairs for each memory node with templates composed by human annotators, combined with 1K manual paraphrasing steps. More details for dataset collection are provided in Section 3.

2 Method

Figure 2 illustrates the overall architecture and the model components that make up the Episodic Memory QA Net. We examine each module in detail and provide our rationale about its formulation in the following sections.

Input Module (Section 2.1): For a given query $\mathbf{q}$, its relevant memory nodes $\mathbf{m} = \{\mathbf{m}^{(k)}\}_{k=1}^{K}$ for slot size K are given as initial memory slots. At test time, relevant memory nodes can be retrieved from a graph search engine that measures textual similarity (*e.g.* n-gram TF-IDF) between its connected node contexts and query. The Query Encoder then encodes the input query with a language model, and the Memory Encoder encodes each memory slot for both structural and semantic properties of each memory.

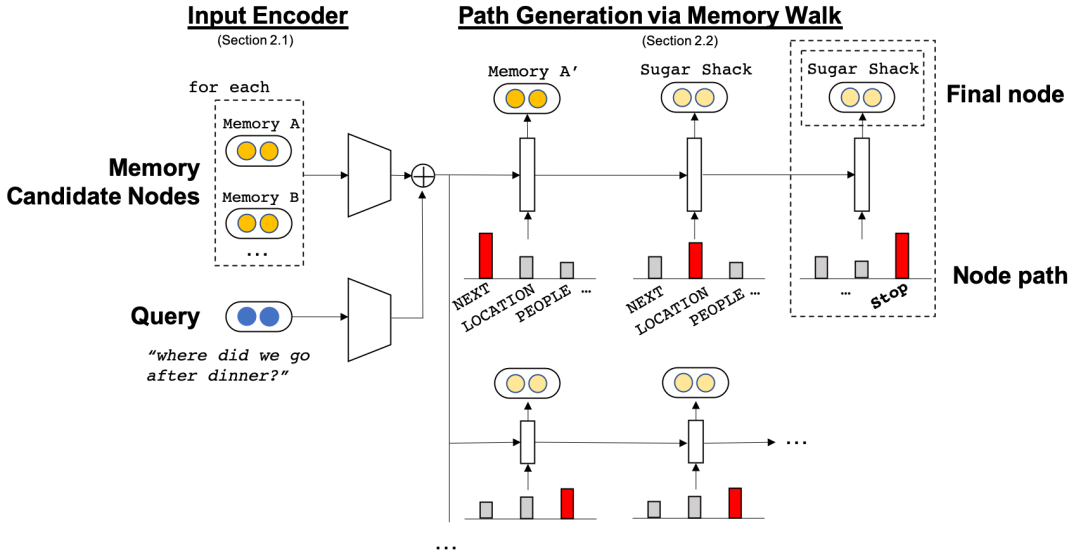


Figure 3: **Memory Graph Network (MGN) Walker**. Input query $\mathbf{q}$ and candidate memory nodes $\mathbf{m} = \{\mathbf{m}^{(k)}\}$ are encoded with the query encoder and the memory encoder, respectively (left). The decoder (right) predicts both the optimal paths and the final nodes $\mathbf{p} = \{\mathbf{p}_t\}_{t=1}^{T}$ based on their relevance scores as well as soft-attention based walk paths, which expands the initial memory slots.

Memory Graph Networks (MGN) (Section 2.2): Many previous work in QA or MRC systems use memory networks to evaluate multiple answer candidates with transitive reasoning, and typically store all potentially relevant raw sentences or bag-of-symbols as memory slots. However, naive increase of memory slot size or retention-based sequential update of memory slots often increase search space for answer candidates, leading to poor precision especially for the Episodic Memory QA task. To overcome this issue, with MGN we store memory graph nodes as initial memory slots, where additional contexts and answer candidates can be succinctly expanded and reached via graph traversals. For each $(\mathbf{q}, \mathbf{m}^{(k)})$ pair, MGN predicts optimal memory slot expansion steps: $\mathbf{p^{(k)}} = \{[\mathbf{p}_{e,t}^{(k)}; \mathbf{p}_{n,t}^{(k)}]\}_{t=1}^{T}$ for edge paths $\mathbf{p}_e$ and corresponding node paths $\mathbf{p}_n$ (Figure 3).

QA Modules (Section 2.3, 2.4): An estimated answer $\hat{\mathbf{a}} = \text{QA}(\mathbf{m}, \mathbf{q})$ is predicted given a query and MGN graph path output from initial memory slots. Specifically, the model outputs a module program $\{\mathbf{u}^{(k)}\}$ for several module networks (*e.g.* `CHOOSE`, `COUNT`, ...) via module selector, each of which produces an answer vector . The aggregated result of module network outputs determines the top-k answers.

2.1 Input Encoding

Query encoder: We represent each textual query with an attention-based Bi-LSTM language model (Conneau et al., 2017) with GloVe (Pennington et al., 2014) distributed word embeddings trained on the Wikipedia and the Gigaword corpus with a total of 6B tokens.

Memory encoder: We represent each memory node based on both its structural features (graph embeddings) and contextual multi-modal features from its neighboring nodes (*e.g.* attribute values).

Structural contexts of each memory node ($\mathbf{m}_s$) are encoded via graph embeddings projection approaches (Bordes et al., 2013), in which nodes with similar relation connectivity are mapped closer in the embeddings space. The model for obtaining embeddings from a MG (composed of subject-relation-object (s, r, o) triples) can be formulated as follows:

$$P(\mathbb{I}_r(s,o)=1|\theta) = \text{score}\big(\mathbf{e}(s), \mathbf{e}_r(r), \mathbf{e}(o)\big) \quad (1)$$

where $\mathbb{I}_r$ is an indicator function of a known relation r for two entities (s,o) (1: valid relation, 0: unknown relation), $\mathbf{e}$ is a function that extracts embeddings for entities, $\mathbf{e}_r$ extracts embeddings for relations, and score$(\cdot)$ is a function (*e.g.* multi-layer perceptrons) that produces a likelihood of a valid triple.

For contextual representation of memories ($\mathbf{m}_c$), we compute attention-weighted sum of textual representation of neighboring nodes and attributes (connected via $\mathbf{r}_j \in \mathbf{R}$), using the same

language model as the query encoder:

$$\mathbf{m}_c = \sum \gamma_j \mathbf{m}_{c,j}$$
$$\gamma = \sigma(\mathbf{W}_{q\gamma}\mathbf{q})$$

Note that the query attention vector γ attenuates or amplifies each attribute of memory based on a query vector to better account for query-memory compatibility accordingly. We then concatenate the structural features with semantic contextual features to obtain the final memory representation ($\mathbf{m} = [\mathbf{m}_s; \mathbf{m}_c]$).

2.2 Memory Graph Network

Inspired by the recently introduced graph traversal networks (Moon et al., 2019) which output discrete graph operations given input contexts, we formulate our MGN as follows. Given a set of initial memory slots ($\mathbf{m}$) and a query ($\mathbf{q}$), the MGN model outputs a sequence path of walk steps ($\mathbf{p}$) within MG to attend to relevant nodes or expand initial memory slots (Figure 3):

$$\{\mathbf{p}^{(k)}\} = \text{MGN}(\mathbf{q}, \{\mathbf{m}^{(k)}\}) \tag{2}$$

Specifically, we define the attention-based graph decoder model which prunes unattended paths, which effectively reduce the search space for memory expansion. We formulate the decoding steps for MGN as follows (bias terms for gates are omitted for simplicity of notation):

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1}) \\
\mathbf{c}_t &= (1 - \mathbf{i}_t) \odot \mathbf{c}_{t-1} \\
&\quad + \mathbf{i}_t \odot \tanh(\mathbf{W}_{zc}\mathbf{z}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1}) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{zo}\mathbf{z}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t) \\
\mathbf{h}_t &= \text{WALK}(\overline{\mathbf{x}}, \mathbf{z}_t) = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{3}$$

where $\mathbf{z}_t$ is a context vector at decoding step t, produced from the attention over graph relations which is defined as follows:

$$\begin{aligned}
\overline{\mathbf{x}} &= \mathbf{W}_{qmx}[\mathbf{q}; \mathbf{m}^{(k)}] \\
\alpha_t &= \sigma(\mathbf{W}_{h\alpha}\mathbf{h}_{t-1} + \mathbf{W}_{x\alpha}\overline{\mathbf{x}}) \\
\mathbf{z}_t &= \mathbf{h}_{t-1} + \sum_{\mathbf{r}_j \in \mathbf{R}} \alpha_{t,j}\mathbf{r}_j
\end{aligned} \tag{4}$$

where $\alpha_t \in \mathbb{R}^{|\mathbf{R}|}$ is an attention vector over the relations space, $\mathbf{r}_k$ is relation embeddings, and $\mathbf{z}_t$ is a resulting node context vector after walking from its previous node on an attended path.

The graph decoder is trained with the ground-truth walk paths by computing the combined loss of $\mathcal{L}_{\text{walk}}(\mathbf{m}, \mathbf{q}, \mathbf{p}) = \sum_{i,t} \mathcal{L}_e + \mathcal{L}_n$ between predicted paths and each of $\{\mathbf{p}_e, \mathbf{p}_n\}$, respectively ($\mathcal{L}_e$: loss for edge paths, and $\mathcal{L}_n$ for node paths):

$$\begin{aligned}
&\sum_{\tilde{\mathbf{p}_e} \neq \mathbf{p}_{e,t}^{(i)}} \max[0, \tilde{\mathbf{p}_e} \cdot \mathbf{p}_{e,t}{}^{(i)} - \alpha_t \mathbf{r} \cdot (\mathbf{p}_{e,t}^{(i)} - \tilde{\mathbf{y}_e})^\top] \\
+ &\sum_{\tilde{\mathbf{p}_n} \neq \mathbf{p}_{n,t}^{(i)}} \max[0, \tilde{\mathbf{p}_n} \cdot \mathbf{p}_{n,t}{}^{(i)} - \mathbf{h}_t{}^{(i)} \cdot (\mathbf{p}_{n,t}^{(i)} - \tilde{\mathbf{p}_n})^\top]
\end{aligned}$$

At test time, we expand the memory slots by activating the nodes along the optimal paths based on the sum of their relevance scores (left) and soft-attention-based output path scores (right) at each decoding step:

$$\mathbf{p}_{n,t}^{(k)} = \underset{\mathbf{p}_n^{(k)} \in \mathbf{V}_{R,1}(\mathbf{p}_{n,t-1}^{(k)})}{\text{argmax}} \mathbf{h}_t \cdot \mathbf{p}_n^{(i)\top} + \sum \alpha_{t,j}\mathbf{r}_j \cdot \mathbf{p}_e^{(k)\top} \tag{5}$$

2.3 Module Networks

MGN outputs are then passed to module networks for the final stage of answer prediction. We extend the previous work in module networks (Kottur et al., 2018), often used in VQA tasks, to accommodate for graph nodes output via MGN. We first formulate the module selector which outputs the module label probability $\{\mathbf{u}^{(k)}\}$ given input contexts for each memory node, trained with cross-entropy loss $\mathcal{L}_{\text{module}}$:

$$\{\mathbf{u}^{(k)}\} = \text{Softmax}(\text{MLP}(\mathbf{q}, \{\mathbf{m}^{(k)}\})) \tag{6}$$

We then define the memory attention to attenuate or amplify all activated memory nodes based on their compatibility with query, formulated as follows:

$$\beta = \text{MLP}(\mathbf{q}, \{\mathbf{m}^{(k)}\}, \{\mathbf{p}^{(k)}\}) \tag{7}$$
$$\alpha = \text{Softmax}(\mathbf{W}_\beta^\top \beta) \in \mathbb{R}^K \tag{8}$$

For this work, we propose the following four modules: `CHOOSE`, `COUNT`, `CONFIRM`, `SET_OR`, and `SET_AND`, hence $\mathbf{u}^{(k)} \in \mathbb{R}^5$. Note that the formulation can be extended to the auto-regressive decoder in case sequential execution of modules is required.

`CHOOSE` module outputs answer space vector by assigning weighted sum scores to nodes along the MGN soft-attention walk paths. End nodes with the most probable walk paths thus get the

highest scores, and their node attribute values are considered as answer candidates. `COUNT` module counts the query-compatible among the activated nodes, $\mathbf{a} = \mathbf{W}_K^\top([\alpha; \max\{\alpha\}; \min\{\alpha\}])$. `CONFIRM` uses a similar approach to `COUNT`, except it outputs a binary label indicating whether the memory nodes match the query condition: $\mathbf{a} = \mathbf{W}_b^\top([\alpha; \max\{\alpha\}; \min\{\alpha\}])$. `SET` modules either combine or find intersection among answer candidates by updating the answer vectors with $\mathbf{a} = \max\{\mathbf{W}_s^\top\{\mathbf{a}^{(k)}\}\}$ or $\mathbf{a} = \min\{\mathbf{W}_s^\top\{\mathbf{a}^{(k)}\}\}$.

2.4 Answer Decoding

Answers from each module network (Section 2.3 are then aggregated as weighted sum of answer vectors with module probability (Eq.6), guided by memory attention (Eq.7). Predicted answers are evaluated with cross-entropy loss $\mathcal{L}_{\text{ans}}$

We observe that the model performs better when the MGN component of the model is pre-trained with ground-truth paths. We thus first train the MGN network with the same training split (without answer labels), and then train the entire model with module networks, fully end-to-end supervised with $\mathcal{L} = \mathcal{L}_{\text{walk}} + \mathcal{L}_{\text{module}} + \mathcal{L}_{\text{ans}}$.

3 Dataset

To empirically evaluate the proposed approach for the Episodic Memory QA task, we create a new dataset, *MemQA*, of 100K question and answer pairs composed based on synthetic memory graphs that are artificially generated. We specifically use the synthetically generated MG to avoid the need for inferring memory graphs from other structured data (such as publicly available photo albums such as Flicker, etc.) which are often limited in size and domains. We bootstrap our large-scale realistic memory graph dataset with the following procedures: first, we construct a synthetic social graph with a set number of artificial users, each with randomly generated interest embeddings. We then create a realistic memory graph by randomly choosing participants within the synthetic social graph as well as activities and associated entities from the curated list (of locations, events, public entities, etc.), which is a subset of common-fact Freebase knowledge base (Bast et al., 2014). Each generated memory node thus has connections to entities appearing in KGs, comprising the memory graph together. Finally, given a set of reference memory nodes and neighboring attributes, we pragmatically generate target ground-truth answer samples (*e.g.* single-hop / multi-hop node value, count, set comparison, yes/no, etc.) We then collect QA pairs for each sample with templates composed with human annotators, which are combined with 1K manual paraphrasing steps for added variety and confirmation. We randomly split the QA corpus into into train (70%), validation (15%), and test sets (15%).

Answer Type	%	**Examples**
Location	18	Mt. Rainier, AMC Theater
KG entities & events	17	Iron Man, Coachella
Common nouns, etc.	16	skiing, movie
Person / Group	16	Mark, Jon
Count	15	zero, three
Date / Time	10	01-02-2018, 7 PM
Yes/No	6	-
Miscellaneous	2	-

Table 1: Types of Answers in $MemQA$

4 Empirical Evaluation

Task: Given a query and a set of initial memory graph nodes via graph search, we evaluate the model on the open-ended question answering prediction task.

4.1 Baselines

We choose as baselines the following QA systems (see Section 5 for details), and modify them accordingly to make comparisons with our task:

- MemN2N (Sukhbaatar et al., 2016): uses the end-to-end memory networks with the static set of initial memory slots. Each memory slot is represented with a bag-of-symbols for surrounding attributes and nodes. We use a single Softmax layer for answer classification. Memory slot size is tuned as a hyper-parameter.

- MemexNet (Jiang et al., 2018): uses the textual representation for multi-modal attributes, and frames the problem into a classification problem via text kernel match approaches to predict approximate answers. Since the dataset does not contain images, we omit the CNN representations.

We also consider several configurations of our proposed approach to examine contributions of

Components	Model	Precision@k			
		k=1	3	5	10
A	`MemN2N` (Sukhbaatar et al., 2016)	29.7	43.8	51.0	50.5
A	`Memex` (Jiang et al., 2018)	32.6	50.3	58.2	59.1
G	`MemQANet` (ablation)	36.7	56.8	63.5	67.9
G + N	`MemQANet` (ablation)	41.1	65.8	75.5	80.0
G + N + E	`MemQANet` (**proposed**)	**45.8**	**68.1**	**75.7**	**80.9**

Table 2: QA performance on the *MemQA* dataset (metric: precision@k). Our proposed model is compared against the selected baseline models as well as several ablation variations of the proposed model (model components (G): Memory Graph Network, (N): Module Networks, (E): graph embeddings).

each component (model components (G): Memory Graph Network, (N): Module Networks, (E): structural graph embeddings).

- (**Proposed**; G+N+E): is the proposed approach as described in Figure 2.
- (G+N): does not structural graph embeddings for MGN, and relies on semantic representation of surrounding nodes.
- (G): does not use any of the module networks (*e.g.* `COUNT`, ...), and predicts MGN graph output as answers instead.

4.2 Results

Parameters: We tune the parameters of each model with the following search space (bold indicate the choice for our final model): graph embeddings size: {64, **128**, 256, 512}, Bi-LSTM hidden states for the language model: {64, **128**, 256, 512}, MGN hidden states: {64, **128**, 256, 512}, word embeddings size: {100, 200, **300**}, and max memory slots: {1, 5, **10**, 20, 40, 80}. We optimize the parameters with Adagrad (Duchi et al., 2011) with batch size 10, learning rate 0.01, epsilon 10^{-8}, and decay 0.1.

Main Results: Table 2 shows the results of the top-k predictions of the proposed model and the baselines. It can be seen that the proposed Memory QA model outperforms other QA baselines for precision at all ks.

Specifically, with the MGN walker model, the MemQA model learns to condition its walk path on query contexts and attend and expand memory nodes, thus outperforming the baseline models that simply rely on their initial memory slots, typically large in size to maintain reasonable recall. The node expansion via MGN allows the model to keep the initial memory slots small (10) and expand only when necessary (*e.g.* for queries that require references to related memories, *"... where did I go after ..."*), thus improving the precision performance. Note that memory slot sizes for baselines are tuned for their performance on the validation set.

In addition, it can be seen that the neural module components (G+N and G+N+E) greatly outperform the ablation model (G) and the baselines by aggregating answers with the modules specifically designed for various types of questions. These neural modules allow the model to answer questions that are typically hard to answer (*e.g.* count, set comparison, etc.) by explicitly reducing the answer space accordingly.

Note also that the graph embeddings (G+N+E) improve the performance over the ablation model that does not use structural contexts (G+N), indicating that the model learns to better leverage knowledge graph contexts to answer questions.

Error Analysis: Table 3 shows some of the example output from the proposed model, given the input question and memory graph nodes. It can be seen that the model is able to predict answers by combining answer contexts from multiple components (walk path, node attention, neural modules, etc.) In general, the MGN walker successfully explores the respective single-hop or multi-hop relations within the memory graph, while keeping the initial memory slots small enough. The activated nodes via graph traversals are then used as input for each neural module, the aggregated results of which are the final top-k answer predictions. There are some cases where the final answer prediction is incorrect, whereas its walk path is correctly predicted. This is due to inaccurate prediction of memory attention vector given a query and initial memory slots, which requires compre-

Question and Answer	Model Preidction		
	Walk Path	Neural Module	Top-k Answers
Q: *Where did Jon and I go after we watched* Avengers? // **A**: *Symphony Hall*	$m_1 \rightarrow$ (NEXT_ID) $\rightarrow m_1'$ $\rightarrow$ (LOCATION)	CHOOSE	Symphony Hall, AMC Theatre, ...
Q: *Who did I go skiing with last year?* **A**: {*Emma, Jacob*}	$m_1 \rightarrow$ (PARTICIPANT) $m_3 \rightarrow$ (PARTICIPANT)	SET_OR	{Emma, Jacob}, {Emma, Noah}, ...
Q: *How many sci-fi movies have I watched* last year? // **A**: *Two*	$m_1 \rightarrow$ (ENTITY) $\rightarrow e_1'$ $\rightarrow$ (GENRE) $\rightarrow a_1'$	COUNT	Three, Two, Four, ...

Table 3: **Error Analysis**: Model predictions of walk paths (with expanded memory nodes noted with ') are partially shown for each question and ground-truth answer pair. The full reference memory graphs are not shown here due to space constraints. Initial memory slot size = 10.

hensive understanding of surrounding knowledge nodes in the context of the query.

5 Related Work

Memory Networks: Weston et al. (2014); Sukhbaatar et al. (2016) propose Memory Networks with explicit memory slots to contain auxiliary multi-input, now widely used in many QA and MRC tasks for its transitive reasoning capability. Traditional limitations are that memory slots for storing answer candidates are fixed in size, and naively increasing the slot size typically decreases the precision. Several work extend this line of research, for example by allowing for dynamic update of memory slots given streams of input (Kumar et al., 2016; Tran et al., 2016; Xu et al., 2019), reinforcement learning based retention control (Jung et al., 2018), etc. By allowing for storing graph nodes as memory slots and for slot expansion via graph traversals, our proposed Memory Graph Networks (MGN) effectively bypass the issues.

Structured QA systems: often answer questions based on large-scale common fact knowledge graphs (Bordes et al., 2015; tau Yih et al., 2015; Xu et al., 2016; Jain, 2016; Yin et al., 2016; Dubey et al., 2018), typically via an entity linking system and a QA model for predicting graph operations through template matching approaches, etc. Our approach is inspired by this line of work, and we utilize the proposed module networks and the MGN walker model to address unique challenges to Episodic Memory QA.

Machine Reading Comprehension (MRC) systems: aim at predicting answers given evidence documents, typically in length of a few paragraphs (Seo et al., 2017; Rajpurkar et al., 2016, 2018; Cao et al., 2019; tau Yih et al., 2015). Several recent work address multi-hop reasoning within multiple documents (Yang et al., 2018; Welbl et al., 2018; Bauer et al., 2018; Clark et al., 2018) or conversational settings (Choi et al., 2018; Reddy et al., 2018), which require often complex reasoning tools. Unlike in MRC systems that typically rely on language understanding, we effectively utilize structural properties of memory graph to traverse and highlight specific attributes or nodes that are required to answer questions.

Visual QA systems: aim to answer questions based on contexts from images (Antol et al., 2015; Wang et al., 2018; Wu et al., 2018). Recently, neural modules (Kottur et al., 2018) are proposed to address specific challenges to VQA such as visual co-reference resolutions, etc. Our work extends the idea of neural modules for Episodic Memory QA by implementing modules that can take graph paths as input for answer decoding. Jiang et al. (2018) proposes visual memex QA which tackles similar problem domains given a dataset collected around photo albums. Instead of relying on meta information and multi-modal content of a photo album, our work explicitly utilizes semantic and structural contexts from memory and knowledge graphs. Another recent line of work for VQA includes graph based visual learning (Hudson and Manning, 2019), which aims to represent each image with a sub-graph of visual contexts. While graph-based VQA operates on a graph constructed from a single scene, Episodic Memory QA operates on a large-scale memory graph with knowledge nodes. We therefore propose memory graph networks to handle ambiguous candidate nodes, a main contribution of the proposed work.

6 Conclusions

We introduce Episodic Memory QA, the task of answering personal user questions grounded on memory graph (MG). The dataset is generated with synthetic memory graphs with simulated attributes, and accompanied with 100K QA pairs composed via bootstrapped scripts and manual annotations. Several novel model components are proposed for unique challenges for the Episodic Memory QA: 1) Memory Graph Networks (MGN) extends the conventional memory networks by enabling dynamic expansion of memory slots through graph traversals, which also naturally allows for explainable predictions. 2) Several neural module networks are proposed for the proposed task, each of which takes queries and memory graphs as input to infer answers. 3) The main Episodic Memory QA Net aggregates answer prediction from each neural module to generate final answer candidates. The empirical results demonstrate the efficacy of the proposed model in the Memory QA reasoning.

References

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *ICCV*.

Hannah Bast, Florian Baurle, Bjorn Buchhold, and Elmar Haussmann. 2014. Easy access to the freebase dataset. In *WWW*.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. 2018. Commonsense for generative multi-hop question answering tasks. *EMNLP*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory network. *arxiv*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2019. Question answering by reasoning across documents with graph convolutional networks. *NAACL*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *EMNLP*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. 2018. Earl: Joint entity and relation linking for question answering over knowledge graphs. *ESWC*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*.

Drew A Hudson and Christopher D Manning. 2019. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *CVPR*.

Sarthak Jain. 2016. Question answering over knowledge base using factual memory networks. *NAACL*.

Lu Jiang, Junwei Liang, Liangliang Cao, Yannis Kalantidis, Sachin Farfade, and Alexander Hauptmann. 2018. Memexqa: Visual memex question answering. *arXiv*.

Hyunwoo Jung, Moonsu Han, Minki Kang, and Sungju Hwang. 2018. Learning what to remember: Long-term episodic memory networks for learning from streaming data. *arXiv preprint arXiv:1812.04227*.

Satwik Kottur, José MF Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2018. Visual coreference resolution in visual dialog using neural module networks. In *ECCV*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv:1606.05250*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2018. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *ICLR*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2016. End-to-end memory networks. *NIPS*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. *NAACL*.

Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, and Anton van den Hengel. 2018. Fvqa: Fact-based visual question answering. *PAMI*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Qi Wu, Chunhua Shen, Peng Wang, Anthony Dick, and Anton van den Hengel. 2018. Image captioning and visual question answering based on attributes and external knowledge. *PAMI*.

Kun Xu, Yuxuan Lai, Yansong Feng, and Zhiguo Wang. 2019. Enhancing key-value memory neural networks for knowledge based question answering. *NAACL*.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. *ACL*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *EMNLP*.

Wen tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. *ACL*.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. *COLING*.

TripleNet: Triple Attention Network for Multi-Turn Response Selection in Retrieval-based Chatbots

Wentao Ma[†], **Yiming Cui**[‡†], **Nan Shao**[†],
Su He[†], **Wei-Nan Zhang**[‡], **Ting Liu**[‡], **Shijin Wang**[†§], **Guoping Hu**[†]
[†]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[‡]Research Center for Social Computing and Information Retrieval (SCIR),
Harbin Institute of Technology, Harbin, China
[§]iFLYTEK AI Research (Hebei), Langfang, China
[†§]{wtma,ymcui,nanshao,suhe,sjwang3,gphu}@iflytek.com
[‡]{ymcui,wnzhang,tliu}@ir.hit.edu.cn

Abstract

We consider the importance of different utterances in the context for selecting the response usually depends on the current query.[1] In this paper, we propose the model TripleNet to fully model the task with the triple $\langle context, query, response \rangle$ instead of $\langle context, response \rangle$ in previous works. The heart of TripeNet is a novel attention mechanism named triple attention to model the relationships within the triple at four levels. The new mechanism updates the representation for each element based on the attention with the other two concurrently and symmetrically. We match the triple $\langle C, Q, R \rangle$ centered on the response from char to context level for prediction. Experimental results on two large-scale multi-turn response selection datasets show that the proposed model can significantly outperform the state-of-the-art methods. [2]

1 Introduction

To establish a human-machine dialogue system is one of the most challenging tasks in Artificial Intelligence (AI). Existing works on building dialogue systems are mainly divided into two categories: retrieval-based method (Yan et al., 2016; Zhou et al., 2016), and generation-based method (Vinyals and Le, 2015). The retrieval-based method retrieves multiple candidate responses from the massive repository and selects the best one as the system's response, while the generation-based method uses the encoder-decoder framework to generate the response, which is similar to machine translation.

A: i downloaded angry ip scanner and now it doesn't work and i can't **uninstall** it
B: you **installed** it via package or via some binary installer
A: i **installed** from ubuntu soft center
B: hm i do n't know what package it is but it should let you remove it the same way
A: ah makes sense then ... hm was it a deb **file**

True Response: i think it was another **format** mayge sth starting with r
False Response: thanks i appreciate it try sudo apt-get **install** libxine-extracodecs

Figure 1: A real example in the Ubuntu Corpus. The upper part is the conversation between speaker A and B. The speaker A want to uninstall the ip scanner and the current query is about the format of the package, so the true response is about the format, but the existing conversation model can be easily misled by the high frequency term 'install' as they deal with the query and other utterances in the same way.

In this paper, we are focusing on the retrieval-based method because it is more practical in applications. Selecting a response from a set of candidates is an important and challenging task for the retrieval-based method. Many of the previous approaches are based on Deep Neural Network (DNN) to select the response for single-turn conversation (Lu and Li, 2013). We study multi-turn response selection in this paper, which is rather difficult because it not only requires identification of the important information such as keywords, phrases, and sentences, but also the latent dependencies between the context, query, and candidate response.

Previous works (Zhou et al., 2018; Wu et al., 2017) show that representing the context at different granularities is vital for multi-turn response selection. However, it is not enough for multi-turn response selection. Figure 1 illustrates the problem with a real example in Ubuntu Corpus. As demonstrated, the following two points should be

[1] In this paper, we define the last message which is waiting for a response as the 'query,' the conversation history including the query as 'context,' and each message in the context as an 'utterance.'

[2] TripleNet source code is available at https://github.com/wtma/TripleNet.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 737–746
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

modeled to solve the problem: (1) the importance of current query should be highlighted, because it has great impact on the importance of different utterances in the context. For example, the query in the case is about the format of the file ('deb file'), which leads the last two utterances (including the query) are more important than the previous ones. If we only match the response with the context, the model may be misled by the high frequency word 'install' and choose the false candidate. (2) the information of different granularities is important, which includes not only the word, utterance, and context level, but also the char level. For example, the different tenses ('install,' 'installed') and the misspelling word ('angry') appear constantly in the conversation. Similar to the role of question for the task of machine reading comprehension (Seo et al., 2016; Cui et al., 2017; Chen et al., 2019), the query in this task is also the key to selecting the response. In this paper, we propose a model named TripleNet to excavate the role of query. The main contributions of our work are listed as follows.

- we use a novel triple attention mechanism to model the relationships within $\langle C, Q, R\rangle$ instead of $\langle C, R\rangle$;
- we propose a hierarchical representation module to fully model the conversation from char to context level;
- The experimental results on Ubuntu and Douban corpus show that TripleNet significantly outperform the state-of-the-art result.

2 Related Works

Earlier works on building the conversation systems are generally based on rules or templates (Walker et al., 2001), which are designed for the specific domain and need much human effort to collect the rules and domain knowledge. As the portability and coverage of such systems are far from satisfaction, people pay more attention to the data-driven approaches for the open-domain conversation system (Ritter et al., 2011; Higashinaka et al., 2014). The main challenge for open-domain conversation is to produce a corresponding response based on the current context. As mentioned previously, the retrieval-based and generation-based methods are the mainstream approaches for conversational response generation. In this paper, we focus on the task response selection which belongs to retrieval-based approach.

The early studies of response selection generally focus on the single-turn conversation, which use only the current query to select the response (Lu and Li, 2013; Ji et al., 2014; Wang et al., 2015). Since it is hard to get the topic and intention of the conversation by single-turn, researchers turn their attention to multi-turn conversation and model the context instead of the current query to predict the response. First, Lowe et al. (2015) released the Ubuntu Dialogue dataset and proposed a neural model which matches the context and response with corresponding representations via RNNs and LSTMs. Kadlec et al. (2015) evaluate the performances of various models on the dataset, such as LSTMs, Bi-LSTMs, and CNNs. Later, Yan et al. (2016) concatenated utterances with the reformulated query and various features in a deep neural network. Baudiš et al. (2016) regarded the task as sentence pair scoring and implemented an RNN-CNN neural network model with attention. Zhou et al. (2016) proposed a multi-view model with CNN and RNN, modeling the context in both word and utterance view. Further, Xu et al. (2017) proposed a deep neural network to incorporate background knowledge for conversation by LSTM with a specially designed recall gate. Wu et al. (2017) proposed matching the context and response by their word and phrase representations, which had significant improvement from previous work. Zhang et al. (2018) introduced a self-matching attention to route the vital information in each utterance, and used RNN to fuse the matching result. Zhou et al. (2018) used self-attention and cross-attention to construct the representations at different granularities, achieving a state-of-the-art result.

Our model is different from the previous methods: first we model the task with the triple $\langle C, Q, R\rangle$ instead of $\langle C, R\rangle$ in the early works, and use a novel triple attention matching mechanism to model the relationships within the triple. Then we represent the context from low (character) to high (context) level, which constructs the representations for the context more comprehensively.

3 Model

In this section, we will give a detailed introduction of the proposed model TripleNet. We first formal-

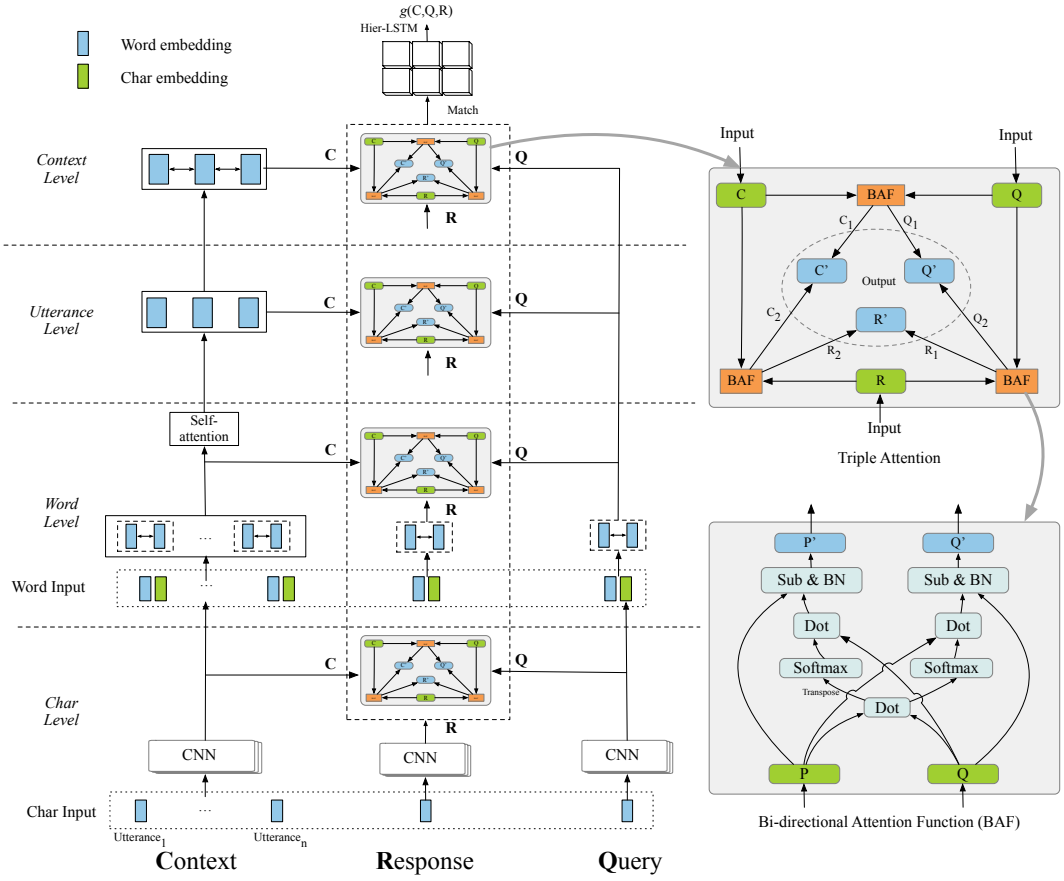


Figure 2: The neural architecture of the model TripleNet. *(best viewed in color)*

ize the problem of the response selection for multi-turn conversation. Then we briefly introduce the overall architecture of the proposed model. Finally, the details of each part of our model will be illustrated.

3.1 Task Definition

For the response selection, we define the task as given the context C, current query Q and candidate response R, which is different from almost all the previous works (Zhou et al., 2018; Wu et al., 2017). We aim to build a model function $g(C, Q, R)$ to predict the possibility of the candidate response to be the correct response.

$$score = g(C, Q, R) \tag{1}$$

The information in context is composed of four levels: context, utterances, words and characters, which can be formulated as $C = (u_1, u_2, ..., u_i, ..., u_n)$, where u_i represents the ith utterance, and n is the maximum utterance number. The last utterance in the context is query $Q = U_n$; we still use query as the end of context to maintain the integrity of the information in context. Each utterance can be formulated as $u_i = (w_1, ..., w_j, .., w_m)$, where w_j is the jth word in the utterance and m is the maximum word number in the utterance. Each word can be represented by multiply characters $w_j = (ch_1, ..., ch_k, .., ch_l)$, where ch_k is the kth char and l is the length of the word in char-level. The latter two levels are similar in the query and response.

3.2 Model Overview

The overall architecture of the model TripleNet is displayed in Figure 2. The model has a bottom-up architecture that organizes the calculation from char to context level. In each level, we first uses the hierarchical representation module to construct the representations of context, response and query. Then the triple attention mechanism is applied to update the representations. At last, the model matches them while focused on the response and fuses the result for prediction.

In the hierarchical representation module, we represent the conversation in four perspectives including char, word, utterance, and context. In the char-level, a convolutional neural network (CNN)

is applied to the embedding matrix of each word and produces the embedding of the word by convolution and maxpooling operations as the char-level representation. In word-level, we use a shared LSTM layer to obtain the word-level embedding for each word. After that, we use self-attention to encode the representation of each utterance into a vector which is the utterance-level representation. At last, the utterance-level representation of each utterance is fed into another LSTM layer to further model the information among different utterances, forming the context-level representation.

The structure of the triple attention mechanism can be seen in the right part of Figure 2. We first design a bi-directional attention function (BAF) to calculate the attention between two sequences and output their new representations. To model the relationship of the triple $\langle C, Q, R\rangle$, we apply BAF to each pair within the triple and get two new representations for each one element, and then we add them together as its final attention-based representation. In the triple attention mechanism, we can update the representation of each one based on the attention result with the other two simultaneously, and each element participates in the whole calculation in the same way.

3.3 Hierarchical Representation

Char-level Representation. At first, we embed the characters in each word into fixed size vectors and use a CNN followed by max-pooling to get character-derived embeddings for each word, which can be formulated by

$$ch_{j,t} = tanh(W_1^j * x_{t:t+s_j-1} + b_1^j) \quad (2)$$

$$ch_j = MaxPooling_{t=0}^{L}[ch_{j,t}] \quad (3)$$

where W_1^j, b_1^j are parameters, $x_{t:t+s_j-1}$ refers to the concatenation of the embedding of (x_t,...,x_{t+s_j-1}), s_j is the window size of jth filter, and the ch is the representation of the word in char-level.

Word-level Representation. Furthermore, we embed word x by pre-trained word vectors, and we also introduce a word matching (MF) feature to the embedding to make the model more sensitive to concurrent words. If the word appears in the response and context or query simultaneously, we set the feature to 1, otherwise to 0.

$$e(x) = [W_e \cdot x; ch(x); MF] \quad (4)$$

where $e(x)$ to denotes the embedding representation, W_e is the pre-trained word embedding, and $ch(x)$ is the character embedding function. We use a shared bi-directional LSTM to get contextual word representations in each utterance, query, and the response. The representation of each word is formed by concatenating the forward and backward LSTM hidden output.

$$\overleftarrow{h(x)} = \overleftarrow{\text{LSTM}}(e(x)) \quad (5)$$

$$\overrightarrow{h(x)} = \overrightarrow{\text{LSTM}}(e(x)) \quad (6)$$

$$h(x) = [\overleftarrow{h(x)}; \overrightarrow{h(x)}] \quad (7)$$

where $h(x)$ is the representation of the word. We denote the word-level representation of the context as $h_u \in \mathbb{R}^{m*d_w}$ and the response as $h_r \in \mathbb{R}^{m*d_w}$, where d_w is the dimension of Bi-LSTMs. Until now, we have constructed the representations of context, query, and response in char and word level, and we only represent the latter two in these two levels because they don't have such rich contextual information as the context.

Utterance-level Representation. Given the k_{th} utterance $u_k = [h_{u_k}^i]_{i=1}^m$, we construct the utterance-level representation by self-attention (Lin et al., 2017):

$$\alpha_i^k = softmax(W_3 tanh(W_2 h_{u_k}(i)^T)) \quad (8)$$

$$u_k = \sum_{i=1}^{m} h_{u_k}^i \alpha_i^k \quad (9)$$

where $W_2 \in \mathbb{R}^{d*d_w}$, $W_3 \in \mathbb{R}^d$ are trainable weights, d is a hyperparameter, u_k is the utterance-level representation, and α_i^k is the attention weight for the ith word in the kth utterance, which signifies the importance of the word in the utterance.

Context-level Representation. To further model the continuity and contextual information among the utterances, we fed the utterance-level representations into another bi-directional LSTM layer to obtain the representation for each utterance in context perspective.

$$c_k = \text{Bi-LSTM}([u_k]_{k=1}^n) \quad (10)$$

where $c_k \in \mathbb{R}^{d_c}$ is the context-level representation for the kth utterance in the context and d_c is the output size of the Bi-LSTM.

3.4 Triple Attention

In this part, we update the representations of context, query, and response in each level

by triple attention, the motivation of which is to model the latent relationships within $\langle context, query, response \rangle$.

Given the triple $\langle C, Q, R \rangle$, we fed each of its pairs into bi-directional attention function (BAF).

$$C_1, Q_1 = BAF(C, Q) \quad (11)$$
$$C_2, R_1 = BAF(C, Q) \quad (12)$$
$$Q_2, R_2 = BAF(C, R) \quad (13)$$
$$C' = BN(C_1 + C_2) \quad (14)$$
$$Q' = BN(Q_1 + Q_2) \quad (15)$$
$$R' = BN(R_1 + R_2) \quad (16)$$

where BN denotes the batch normalization layer (Ioffe and Szegedy, 2015) which is conducive to preventing vanishing or exploding of gradients. BAF produces the new representations for two sequences (P, Q) by the attention from two directions, which is inspired by Seo et al. (2016). We can formulate it by

$$M_{pq} = P^T tanh(W_3 Q) \quad (17)$$
$$Att_{pq} = softmax(M_{pq}) \quad (18)$$
$$Att_{qp} = softmax(M_{pq}^T) \quad (19)$$
$$P' = P - \tilde{Q}; \;\; \tilde{Q} = Q Att_{pq}; \quad (20)$$
$$Q' = Q - \tilde{P}; \;\; \tilde{P} = P Att_{qp}; \quad (21)$$

where Att_{pq}, Att_{qp} are the attention between P and Q in two directions, P', Q' are the new representations the two sequences (P, Q), and we apply a batch normalization layer upon them too.

We find that the triple attention has some interesting features: (1) triple, the representation for each element in the triple $\langle C, Q, R \rangle$ is updated based on the attention to the other two concurrently; (2) symmetrical, which means each element in the triple plays the same role in the structure because their contents are similar in the whole conversation; (3) unchanged dimension, all the outputs of triple attention has the same dimensions as the inputs, so we can stack multiple layers as needed.

3.5 Triple Matching and Prediction

Triple Matching. We match the triple $\langle C, Q, R \rangle$ in each level with the cosine distance using new representations produced by triple attention. This process focuses on the response because it is our target. For example, in the char-level, we match the triple by

$$\tilde{M}^1_{rc}(i, k, j) = cosine(ch'_r(i), ch'_{u_k}(j)) \quad (22)$$
$$M^1_{rc}(i, k) = \max_{0<j<m} \tilde{M}_1(i, j, k) \quad (23)$$
$$M^1_{rq}(i, j) = cosine(ch'_r(i), ch'_q(j)) \quad (24)$$
$$M_1 = [M^1_{rc}(i, k); M^1_{rq}(i, j)] \quad (25)$$

where ch' is the representation updated by triple attention, $M_1 \in \mathbb{R}^{m*(n+m)}$ is the char-level matching result, the word-level matches the triple in the same way, and the utterance and the context level match the triple without the maxpooling operation. We use M_2, M_3, M_4 as the matching results in the word, utterance and context levels.

Fusion. After obtaining the four-level matching matrix, we use hierarchical RNN to get highly abstract features. Firstly, we concatenate the four matrices to form a 3D cube $M \in \mathbb{R}^{m*(n+m)*4}$ and we use m as one of the matrix in M, which denotes the matching result for one word in response in four levels.

$$M = [M_1; M_2; M_3; M_4] \quad (26)$$
$$\tilde{m} = MaxPooling_{i=0}^{n+m}[\text{Bi-LSTM}(m_i)] \quad (27)$$
$$v = MaxPooling_{j=0}^{m}[\text{Bi-LSTM}(\tilde{m}_j)] \quad (28)$$

Where m_i and $\tilde{m}_j$ are the ith, jth row in the matrix m and $\tilde{m}$. We merge the results from different time steps in the outputs of LSTM by max-pooling operation. Until now, we encode the matching result into a single feature vector v.

Final Prediction. For the final prediction, we fed the vector V into a full-connected layer with sigmoid output activation.

$$g(C, Q, R) = sigmoid(W_4 \cdot v + b_4) \quad (29)$$

where W_4, b_4 are trainable weights. Our purpose is to predict the matching score between the context, query and candidate response, which can be seen as a binary classification task. To train our model, we minimize the cross entropy loss between the prediction and ground truth.

4 Experiments

4.1 Dataset

We first evaluate our model on Ubuntu Dialogue Corpus (Lowe et al., 2015) because it is the largest public multi-turn dialogue corpus which consists of about one million conversations in the specific domain. To reduce the number of unknown words,

	Ubuntu Dialogue Corpus				Douban Conversation Corpus					
	R_2@1	R_{10}@1	R_{10}@2	R_{10}@5	MAP	MRR	P@1	R_{10}@1	R_{10}@2	R_{10}@5
DualEncoder	90.1	63.8	78.4	94.9	48.5	52.7	32.0	18.7	34.3	72.0
MV-LSTM	90.6	65.3	80.4	94.6	49.8	53.8	34.8	20.2	35.1	71.6
Match-LSTM	90.4	65.3	80.4	94.6	49.8	53.8	34.8	20.2	34.8	71.0
DL2R	89.9	62.6	78.3	94.4	48.8	52.7	33.0	19.3	34.2	70.5
Multi-View	90.8	66.2	80.1	95.1	50.5	54.3	34.2	20.2	35.0	72.9
SMN	92.6	72.6	84.7	96.1	52.9	56.9	39.7	23.3	39.6	72.4
RNN-CNN	91.1	67.2	80.9	95.6	-	-	-	-	-	-
DUA	-	75.2	86.8	96.2	*55.1*	59.9	42.1	24.3	*42.1*	*78.0*
DAM	*93.8*	*76.7*	*87.4*	*96.9*	55.0	*60.1*	*42.7*	*25.4*	41.0	75.7
TripleNet	**94.3**	**79.0**	**88.5**	**97.0**	**56.4**	**61.8**	**44.7**	**26.8**	**42.6**	77.8
TripleNet$_{elmo}$	95.1	80.5	89.7	97.6	60.9	65.0	47.0	27.8	48.7	81.4
TripleNet$_{ensemble}$	95.6	82.1	90.9	98.0	63.2	67.8	51.5	31.3	49.4	83.2

Table 1: Experimental results on two public dialogue datasets. The table is segmented into three sections: Non-Attention models, Attention-based models and our models. The italics denotes the previous best results, and the scores in bold express the new state-of-the-art result of single model without any pre-training layer.

we use the shared copy of the Ubuntu corpus by Xu et al. (2017) which replaces the numbers, paths, and URLs by specific symbols.[3] Furthermore, to verify the generalization of our model, we also carry out experiments on Douban Conversation Corpus (Wu et al., 2017), which shares similar format with the Ubuntu corpus but is open-domain and in the Chinese language.

For the Ubuntu corpus, we use the recall at position k in n candidate responses ($R_n@k$) as evaluation metrics, and we use MAP (Mean Average Precision), MRR (Mean Reciprocal Rank), and Precision-at-one as the additional metrics for Douban corpus, following the previous work (Wu et al., 2017).

4.2 Experiment Setup

We implement our model by Keras (Chollet et al., 2015) with TensorFlow backend. In the Embedding Layer, the word embeddings are pre-trained using the training set via GloVe (Pennington et al., 2014), the weights of which are trainable. For char embedding, we set the kernel shape as 3 and filter number as 200 in the CNN layer. For all the Bi-directional LSTM layers, we set their hidden size to 200. We use Adamax (Kingma and Ba, 2014) for weight updating with an initial learning rate of 0.002. For ensemble models, we generate 6 models for each corpus using different random seeds and merge the result by voting.

For better comparison with the baseline models, the main super parameters in TripleNet, such as the embedding size, max length of each turn, and the vocabularies, are the same as those of the baseline models. The maximum number of conversation turns, which changes with the models, is 12 in our model, 9 in DAM (Wu et al., 2017), and 10 in SMN (Wu et al., 2017).

4.3 Baseline Models

We basically divided baseline models into two categories for comparisons.

Non-Attention Models. The majority of the previous works on this task are designed without attention mechanisms, including the Sequential Matching Network (SMN) (Wu et al., 2017), Multi-View model (Zhou et al., 2016), Deep Learning to Respond (DL2R) (Yan et al., 2016), Match-LSTM (Wang and Jiang, 2016), MV-LSTM (Wan et al., 2016), and DualEncoder (Lowe et al., 2015).

Attention-based Models. The attention-based models typically match the context and the candidate response based on the attention among them, including DAM (Zhou et al., 2018), DUA (Zhang et al., 2018), and RNN-CNN (Baudiš et al., 2016).

4.4 Overall Results

The overall results on two datasets are depicted in Table 1. Our results are obviously better on the two datasets compared with recently attention-based model DAM, which exceeds 2.3% in $R_{10}@1$ of Ubuntu and 2.6% in $P@1$ of Douban. Furthermore, our score is significantly exceeding in almost all metrics except the $R_{10}@5$ in Douban when compared with DUA, which may be be-

[3] https://www.dropbox.com/s/2fdn26rj6h9bpvl/ubuntudata.zip

cause the metric is not very stable as the test set in Douban is very small (1000).

To further improve the performance, we utilize pre-trained ELMo (Peters et al., 2018) and fine-tune it on the training set in the Ubuntu condition while we train ELMo from scratch using the Douban training set. As the baseline of Douban corpus is relatively lower, we observe much bigger improvements in the corpus using ELMo. The model ensemble has further improvements based on the single model with ELMo; the score of $R_{10}@1$ in Ubuntu is close to the average performance of human experts at 83.8 (Lowe et al., 2016).

Compared to non-attention models such as the SMN and Multi-view, which match the context and response at two levels, TripleNet shows substantial improvements. On $R_{10}@1$ for Ubuntu corpus, there is a 6.3% absolute improvement from SMN and 12.8% from Multi-view, showing the effectiveness of triple attention.

4.5 Model Ablation

To better demonstrate the effectiveness of TripleNet, we conduct the ablations on the model under the Ubuntu corpus for its larger data size.

We first remove the triple attention and matching parts (-TAM); the result shows a marked decline (2.4% in $R_{10}@1$), which is in the second part of Table 2. The performance of the model is similar to the baseline model DAM. This indicates that our four-level hierarchical representation may play a similar role to the five stacks Transformer in DAM. We then remove the triple attention part, which means we match the pairs $\langle C, R\rangle$ and $\langle Q, R\rangle$ with their original representation in each level; the score of $R_{10}@1$ drops 1.4%, which shows the effect of triple attention. We also have tried to remove all the parts related to the query (-Query). That means the attention and matching parts are only calculated within the pair $\langle C, R\rangle$. It is worth mentioning that the information of the query is still contained at the end of the context. The performance also has a marked drop (1.6% in $R_{10}@1$), which shows that it is necessary to model the query separately. To find out which subsection in those parts is more important, we remove each one of them.

Triple attention matching ablation. As we can see in the third part of Table 2, when attention between context and response is removed (-A_{CR}), the largest decrease (0.6% in $R_{10}@1$) appears, which indicates that the relationship between context and response is most important in the triple. The attentions in the other two pairs $\langle C, Q\rangle$ and $\langle Q, R\rangle$ all lead to a slight performance drop (0.3 and 0.5 in $R_{10}@1$), which may be because they overlap with each other for updating the representation of the triple.

When we remove the matching between context and response, we find that the performance of the model has a marked drop (2.1 in $R_{10}@1$), which shows that the relationship within $\langle C, R\rangle$ is the base for selecting the response. The query and response matching part also leads to a significant decline. This shows that we should pay more attention to query within the whole context.

Hierarchical representation ablation. To find out the calculation of which level is most important, we also tried to remove each level calculation from the hierarchical representation module, which can be seen in the fourth part of Table 2. To our surprise, when we remove char (-char) and context level calculation (-context), we observe that the reduction (0.5 in $R_{10}@1$) is more significant than the other two, indicating that we should pay more attention to the lowest and highest level information. Also by removing the other two levels, there is also a significant reduction from TripleNet, which means each level of the three is indispensable for our TripleNet .

From the experiments in this part, we find that each subsection of the hierarchical representation module only leads to a slight performance drop. Maybe it's because the representation from each

	$R_2@1$	$R_{10}@1$	$R_{10}@2$	$R_{10}@5$
TripleNet	94.3	79.0	88.5	97.0
-TAM	93.5	76.6	86.8	96.6
-A_{tri}	93.8	77.6	87.6	96.9
-Query	93.8	77.4	87.3	96.6
-A_{CR}	94.1	78.4	87.9	97.0
-A_{QR}	94.1	78.5	88.1	97.0
-A_{CQ}	94.3	78.7	88.3	97.0
-M_{CR}	93.7	76.9	87.0	96.7
-M_{QR}	94.4	78.5	88.1	97.1
-char	94.1	78.3	88.0	97.1
-word	94.3	78.5	88.2	97.0
-utterance	94.1	78.6	88.1	97.1
-context	94.0	78.4	88.0	97.0

Table 2: Ablation studies on Ubuntu Dialogue Corpus. The letter 'A' stands for the subsection in triple attention, and 'M' the is triple matching part.

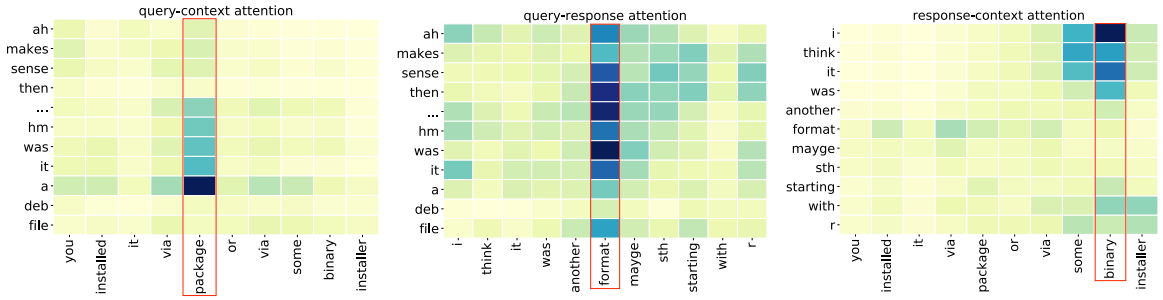


Figure 3: The attention visualization among the query, context, and response in word-level.

level represent the conversation from a unique and indispensable perspective, and the information conveyed by different representations may have some overlap.

5 Analysis and Discussion

5.1 Visualization

By decoding our model for the case in Figure 1, we find that our model TripleNet can choose the true response. To analyze in detail how triple attention works, we get the attention in word-level as the example and visualize it in Figure 3. As there are so many words in the context, we only use the second utterance in the upper part of Figure 1 for its relatively rich semantics.

In the query-context attention, the query mainly pays attention to the keyword 'package.' This is helpful to get the topic of the conversation. While the attention of context focuses on the word 'a' which is near the key phrase 'deb file,' which may be because the representation of the word catches some information from the words nearby by Bi-LSTM. In the query-response attention, the result shows that the attention of the query mainly focuses on the word 'format,' which is the most important word in the response. But we can also find that the response does not catch the important words in the query. In the response-context attention, the response pays more attention to the word 'binary,' which is another important word in the context.

From the three maps, we find that each attention can catch some important information but miss some useful information too. If we join the information in query-context and response-context attention, we can catch the most import information in the context. Furthermore, the query-response attention can help us catch the most important word in the response. So it is natural for TripleNet to select the right response because the model can integrate the three attentions together.

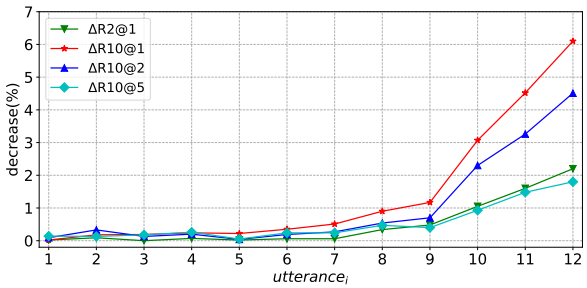


Figure 4: The decrease of the performance when the $utterance_i$ is removed in Ubuntu Corpus.

5.2 Discussion

In this section, we will discuss the importance of different utterances in the context. To find out the importance of different utterances in the context, we conduct an experiment by removing each one of them with the model (-Query) in the ablation experiment part because the model deals all the utterances include the query in the same way. For each experiment in this part, we remove the ith ($0 < i < 13$ and $Q = U_{12}$) utterance in the context both in training and evaluation processes and report the decrease of performance in Figure 4. We find that the removing of the query leads the most significant decline (more than 6% in $R_{10}@1$), that indicates the query is much more important than any other utterances. Furthermore, the decrease is stable before the 9th utterances and raises rapidly in the last 3 utterances. We can deduce that the last three utterances are more important than the other ones.

From the whole result, we can conclude that it's better to model the query separately than deal all of the utterances in the same way for their significantly different importance; we also find that we should pay more attention to the utterances near

the query because they are more important.

6 Conclusion

In this paper, we propose a model TripleNet for multi-turn response selection. We model the context from low (character) to high (context) level, update the representation by triple attention within $\langle C, Q, R \rangle$, match the triple focused on response, and fuse the matching results with hierarchical LSTM for prediction. Experimental results show that the proposed model achieves state-of-the-art results on both Ubuntu and Douban corpus, which ranges from a specific domain to open domain, and English to Chinese language, demonstrating the effectiveness and generalization of our model. In the future, we will apply the proposed triple attention mechanism to other NLP tasks to further testify its extensibility.

Acknowledgement

We would like to thank all anonymous reviewers for their hard work on reviewing and providing valuable comments on our paper. We also would like to thank Yunyi Anderson for proofreading our paper thoroughly. This work is supported by National Key R&D Program of China via grant 2018YFC0832100.

References

Petr Baudiš, Jan Pichl, Tomáš Vyskočil, and Jan Šedivỳ. 2016. Sentence pair scoring: Towards unified framework for text comprehension. *arXiv preprint arXiv:1603.06127*.

Zhipeng Chen, Yiming Cui, Wentao Ma, Shijin Wang, and Guoping Hu. 2019. Convolutional spatial attention model for reading comprehension with multiple-choice questions. *Proceedings of the AAAI. Honolulu, HI*.

François Chollet et al. 2015. Keras. `https://keras.io`.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2017. Attention-over-attention neural networks for reading comprehension. In *Proceedings of the 55th Annual Meeting of ACL (Volume 1: Long Papers)*, pages 593–602. Association for Computational Linguistics.

Ryuichiro Higashinaka, Kenji Imamura, Toyomi Meguro, Chiaki Miyazaki, Nozomi Kobayashi, Hiroaki Sugiyama, Toru Hirano, Toshiro Makino, and Yoshihiro Matsuo. 2014. Towards an open-domain conversational system fully based on natural language processing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 928–939. Dublin City University and Association for Computational Linguistics.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

Rudolf Kadlec, Martin Schmid, and Jan Kleindienst. 2015. Improved deep learning baselines for ubuntu corpus dialogs. *arXiv preprint arXiv:1510.03753*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294. Association for Computational Linguistics.

Ryan Lowe, Iulian V Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. On the evaluation of dialogue systems with next utterance classification. *arXiv preprint arXiv:1605.05414*.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *International Conference on Neural Information Processing Systems*, pages 1367–1375.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*, pages 1532–1543. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT-2018*, pages 2227–2237. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of EMNLP 2011*, pages 583–593. Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Marilyn A Walker, Rebecca Passonneau, and Julie E Boland. 2001. Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems. In *Proceedings of ACL 2001*, pages 515–522. Association for Computational Linguistics.

Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-srnn: Modeling the recursive matching structure with spatial rnn. *arXiv preprint arXiv:1604.04378*.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. *arXiv preprint arXiv:1503.02427*.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with lstm. In *Proceedings of NAACL-HLT*, pages 1442–1451.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of ACL 2017*, pages 496–505. Association for Computational Linguistics.

Zhen Xu, Bingquan Liu, Baoxun Wang, Chengjie Sun, and Xiaolong Wang. 2017. Incorporating loose-structured knowledge into conversation modeling via recall-gate lstm. In *International Joint Conference on Neural Networks*, pages 3506–3513.

Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–64.

Zhuosheng Zhang, Jiangtong Li, Pengfei Zhu, Hai Zhao, and Gongshen Liu. 2018. Modeling multi-turn conversation with deep utterance aggregation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3740–3752.

Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, Dianhai Yu, Hao Tian, Xuan Liu, and Rui Yan. 2016. Multi-view response selection for human-computer conversation. In *Proceedings of EMNLP 2016*, pages 372–381. Association for Computational Linguistics.

Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. 2018. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of ACL 2018*, volume 1, pages 1118–1127.

Relation Module for Non-answerable Prediction on Reading Comprehension

Kevin Huang, Yun Tang, Jing Huang, Xiaodong He, and Bowen Zhou
JD AI Research, Mountain View, CA
{kevin.huang3, yun.tang, jing.huang,
xiadong.he, bowen.zhou}@jd.com

Abstract

Machine reading comprehension (MRC) has attracted significant amounts of research attention recently, due to an increase of challenging reading comprehension datasets. In this paper, we aim to improve a MRC model's ability to determine whether a question has an answer in a given context (e.g. the recently proposed SQuAD 2.0 task). Our solution is a relation module that is adaptable to any MRC model. The relation module consists of both semantic extraction and relational information. We first extract high level semantics as objects from both question and context with multi-head self-attentive pooling. These semantic objects are then passed to a relation network, which generates relationship scores for each object pair in a sentence. These scores are used to determine whether a question is non-answerable. We test the relation module on the SQuAD 2.0 dataset using both the BiDAF and BERT models as baseline readers. We obtain 1.8% gain of F1 accuracy on top of the BiDAF reader, and 1.0% on top of the BERT base model. These results show the effectiveness of our relation module on MRC.

1 Introduction

Ever since the release of many challenging large scale datasets for machine reading comprehension (MRC) (Rajpurkar et al., 2016; Joshi et al., 2017; Trischler et al., 2016; Yang et al., 2018; Reddy et al., 2018; Jia and Liang, 2017), there have been correspondingly many models for these datasets (Yu et al., 2018; Seo et al., 2017; Liu et al., 2018b; Hu et al., 2017; Xiong et al., 2017; Wang et al., 2018; Liu et al., 2018c; Tay et al., 2018). Knowing what you don't know (Rajpurkar et al., 2018) is important in real applications of reading comprehension. Unanswerable questions are commonplace in the real world, and SQuAD 2.0 was released specifically to target this problem (see Figure 1 for an example of non-answerable questions).

Example 1
Context: Each year, the southern California area has about 10,000 earthquakes. Nearly all of them are so small that they are not felt. Only several hundred are greater than magnitude 3.0, and only about 1520 are greater than magnitude 4.0. The magnitude 6.7 1994 **Northridge earthquake** was particularly destructive, causing a substantial number of deaths, injuries, and structural collapses. It caused the most property damage of any earthquake in U.S. history, estimated at over $20 billion.
Question: What earthquake caused $20 million in damage?
Answer: None.

Figure 1: An example of non-answerable question in SQuAD 2.0. Highlighted words are the output from the BERT base model. The true answer is "None".

One problem that most of the early MRC readers have in common is the inability to predict non-answerable questions. Readers on the popular SQuAD dataset have to be modified in order to accommodate a non-answerable possibility. Current methods on SQuAD 2.0 generally attempt to learn a single fully connected layer (Clark and Gardner, 2018; Liu et al., 2018a; Devlin et al., 2018) in order to determine whether a question/context pair is answerable. This leaves out relational information that may be useful for determining answerability. We believe that relationships between different high-level semantics in the context are helpful to make better answerable or unanswerable decision. For example, "Northridge earthquake" is mistakenly taken as the answer to the question about what earthquake caused $20 million in damage. Because "$20 billon" is positioned far away from "Northridge earthquake", it is hard for a model to link these two concepts together and recognize the mismatch of "$20 million" in the question and "$20 billion" in the context.

Motivated by exploiting high level semantic relationships in the context, our first step is to extract meaningful high-level semantics from question/context. Multi-head self-attentive pooling

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 747–756
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

(Lin et al., 2017) has shown to be able to extract different views of a sentence with multiple heads. Each head from the multi-head self attentive pooling has different weights on the context with learned parameters. This allows each head to act as a filter in order to emphasize part of the context. By summing up the weighted context, we obtain a vector representing an instance of a high-level semantic, which we can call it an "object". With multiple heads, we generate different semantic objects, which are then fed in to a relation network.

Relation networks (Santoro et al., 2017) are specifically designed to model relationships between pairs of objects. In the case of reading comprehension, an object would ideally be phrase level semantics within a sentence. Relation networks are able to accomplish modeling these relationships by constraining the network to learn a score for each pair of these objects. After learning all of the pairwise scores, the relation network then summarizes all of the relations to a single vector. By taking a weighted sum of all of the relation scores that the sentence has, we generate a non-answerable score that is trained jointly with answer span scores from any MRC model to determine non-answerability.

In addition, we add in plausible answers from unanswerable examples to help train the relation module. These plausible answers help the base model learn a better span prediction and are also used to help guide our object extractor to extract relevant semantics. We train a separate layer for start-end probabilities based on the plausible answers. We then augment the context vector with hidden states from this layer. This allows the multi-head self-attentive pooling to focus on objects related to the proposed answer span, and differentiate from other objects that are not as relevant in the context.

In summary we propose a new relation module dedicated to learning relationships between high-level semantics and deciding whether a question is answerable. Our contributions are four-fold:

1. Introduce the concept of using multi-head self-attentive pooling outputs as high level semantic objects.

2. Exploit relation networks to model the relationships between different objects in a context. We then summarize these relationships to get a final decision.

3. Introduce a separate feed-forward layer trained on plausible answers so that we can augment the context vector passed into the object extractor. This results in the object extractor extracting phrases more relevant to the proposed answer span.

4. Combining all of the above into a flexible relation module that can be added to the end of a question answering model to boost non-answerable prediction.

To our knowledge, this is the first case of utilizing an object extractor to extract high level semantics, and a relation networks to encode relationships between these semantics in reading comprehension. Our results show improvement on top of the baseline BiDAF model and the state-of-the-art reader based on BERT, on the SQuAD 2.0 task.

2 Related Work

Relation Networks (RN) were first proposed by (Santoro et al., 2017) in order to help neural models to reason over the relationships between two objects. Relation networks learn relationships between objects by learning a pairwise score for each object pair. Relation networks have been applied to CLEVR (Johnson et al., 2017) as well as bAbI (Weston et al., 2015). In the CLEVR dataset, the object inputs to the relation network are visual objects in an image, extracted by a CNN, and in bAbI the object inputs are sentence encodings. In both tasks, the relation network is then used to compute a relationship score over these objects. Relation Networks were further applied to general reasoning by training the model on images (You et al., 2018).

MAC (Memory, Attention and Composition) networks (Hudson and Manning, 2018) are different models that have also been shown to learn relations from the CLEVR dataset. MAC networks operate with read and write cells. Each cell would compute a relation score between a knowledge base and question and write it into memory. Multiple read and write cells are strung together sequentially in order to model long chains of multi-hop reasoning. Although MAC networks do not explicitly reason between pairwise objects as relation networks do, MAC networks are an interesting way of generating multi-hop reasoning between objects within a context.

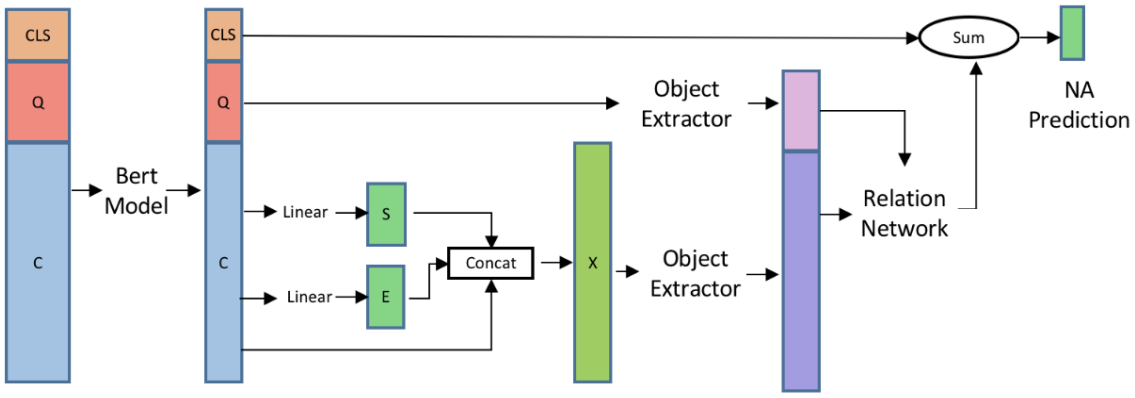


Figure 2: Relation Module on BERT. S and E are hidden states trained by plausible answers. We then concatenate S and E with the contextual representation to feed into the object extractor. After we obtain the extracted objects, we then feed into a Relation Network and pass it down for NA predictions.

Another similar line of work investigated pre-training relationship embeddings across word pairs on large unlabelled corpus (Jameel et al., 2018; Joshi et al., 2018). These pre-trained pairwise relational embeddings were added to the attention layers of BiDAF, where higher level abstract reasoning occurs. The paper showed an impressive gain of 2.7% on the SQuAD 2.0 development set on top of their version of BiDAF.

Many MRC models have been adapted to work on SQuAD 2.0 recently (Hu et al., 2019; Liu et al., 2018a; Sun et al., 2018; Devlin et al., 2018). (Hu et al., 2019) added a separately trained answer verifier for no-answer detection with their Mnemonic Reader. The answer sentence that is proposed by the reader and the question are passed to three combinations of differently configured verifiers for fine-grained local entailment recognition. (Liu et al., 2018a) just added one layer as the unanswerable binary classifier to their SAN reader. (Sun et al., 2018) proposed the U-net with a universal node that encodes the fused information from both the question and passage. The summary U-node, question vector and two context vectors are passed to predict whether the question is answerable. Plausible answers were used for no-answer pointer prediction, while in our approach, plausible answers were used to augment context vector for object extraction that later help the no-answer prediction.

Pretraining embeddings on large unlabelled corpus has been shown to improve many downstream tasks (Peters et al., 2018; Howard and Ruder, 2018; Alec et al., 2018). The recently released BERT (Devlin et al., 2018) greatly increased the F1 scores on the SQuAD 2.0 leaderboard. BERT consists of stacked Transformers (Vaswani et al., 2017), that are pre-trained on vast amounts of unlabeled data with a masked language model. The masked language model helps finetuning on downstream tasks, such as SQuAD 2.0. BERT models contains a special CLS token which is helpful for the SQuAD 2.0 task. This CLS token is trained to predict if a pair of sentences follow each other during the pre-training, which helps encode entailment information between the sentence pair. Due to a strong masked language model to help predict answers and a strong CLS token to encode entailment, BERT models are the current state-of-the art for SQuAD 2.0.

3 Relation Module

Our relation module is flexible, and can be placed on top of any MRC model. We now describe the relation module in detail.

3.1 Augmenting Inputs

Figure 2 shows our relation module on top of the base reader BERT. In addition to the original start-end prediction layers trained from true answers in the base reader, we include a separate start-end prediction layer, with separate parameters, trained specifically on plausible and true answers available in SQuAD 2.0. The context output C from BERT is projected into two hidden state layers S and E, where C, S and $E \in \mathbf{R}^{L \times h}$, L is the context length and h is the hidden size. The S and E layers are then projected down to a hidden

dimension of 1, and trained with Cross-Entropy Loss against the plausible and true answer starts and ends. The hidden states S and E of this layer are concatenated with the last context layer output C and projected back to the original dimension to obtain the augmented context vector X, which is fused with start-end span information.

$$S = tanh(CW_1 + b_1) \quad (1)$$
$$E = tanh(CW_2 + b_2) \quad (2)$$
$$X = [C; S; E]W \quad (3)$$

where [;;] is concatenation of multiple tensors and $X \in \mathbf{R}^{L \times h}$. This process is shown in Figure 2, where S and E are hidden states trained on plausible and true answer spans. This tensor X and the last question layer output Q are passed to the object extractor layer.

3.2 Object Extractor

The augmented context tensor X (and separately, question tensor Q) is passed through the object extractor to generate object representations from the tensor. We pass the inputs through a multi-head self-attentive pooling layer. This object extractor can be thought of as a set filters extracting out areas of interest within a sentence. We multiply the input tensor X with a multi-head self attention matrix A which is defined as

$$A = Softmax(W_4\sigma(W_3X^T)) \quad (4)$$
$$O = AX \quad (5)$$

where $W_3 \in R^{h \times h}$, and $W_4 \in R^{n \times h}$; σ is an activation function, such as $tanh$; n is the number of heads, and h is the hidden dimension. The output $O \in R^{n \times h}$ contains the n objects with hidden dimension h that are passed to the next layer.

3.2.1 Object Extraction Regularization

In order to help encourage the multiple heads to extract different meaningful semantics in the text, a regularization loss (Xia et al., 2018) is introduced to encourage each head to attend to slightly different sections of the context. Overlapping objects centered on the answer span are expected, due to information fused from S and E, but we do not want the entire weight distribution of the head to be solely focused on the answer span. As we show in later figures, many heads heavily weight the answer span, but also weight information relevant to the answer span needed to make a better non-answerable prediction. Our regularization term also helps prevent the multi-headed attentive pooling from learning a noisy distribution over all of the context. This regularization loss is defined as

$$L_{aux} = \alpha||AA^T - I||_2 \quad (6)$$

where A is the weight matrix for the attention heads and I is the identity matrix. α is set to be 0.0005 in our experiments.

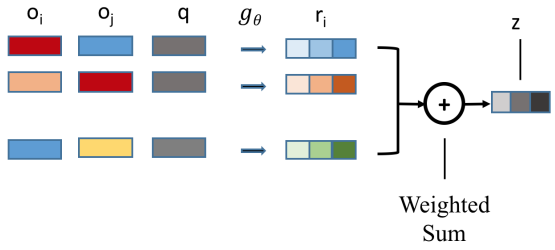


Figure 3: Illustration of a Relation Network. The g_θ is a MLP to score relationships between pairs

3.3 Relation Networks

Extracted objects are subsequently passed to a relation network. We use two layer MLP g_θ (in Figure 3) as a scoring function to compute the similarity between objects. In the question-answering task, the context contains the contextual information necessary to determine whether a question is answerable. Phrases and ideas from various parts of the context need to come together in order to fully understand whether or not a question is answerable. Therefore our relation module takes all pairs of context objects to score, and use the question objects to guide the scoring function. We use 2 question heads q_0, q_1, so our scoring function is:

$$r_i = \sum_{j=0}^{n} \omega_{i,j} * g_\theta(o_i, o_j, q_0, q_1) \quad (7)$$
$$z = \sum_{i=0}^{n} \gamma_i * f_\phi(r_i) \quad (8)$$

where the outputs r_i is the weighted sum of the relation values for object o_i from O, and z is a summarized relation vector. The weights $\Omega_i = [\omega_{i,0}, ..., \omega_{i,n}]$ and $\Gamma = [\gamma_0, ..., \gamma_n]$ are computed by projecting down the relations scores into a hidden size of 1, and applying softmax.

$$\Omega_i = Softmax(g_\theta(o_i, :, q_0, q_1)w_g) \quad (9)$$
$$\Gamma = Softmax(f_\phi(:)w_f) \quad (10)$$

g_θ and f_ϕ are two layer MLP with activation function $tanh$ to compute and aggregate relational scores. Figure 3 shows the process of a single relation network, where two context objects and question objects are passed in to g_θ to obtain the output z.

We project the weighted sum of f_ϕ with a linear layer to a single value as a representation of the non-answerable score. This score is combined with the start/end logits from the base reader, and trained jointly with the reader's cross-entropy loss. By training jointly, the model is able to make a better prediction based on the confidence of the span prediction, as well as the confidence based on the non-answerable score from the relation module.

4 Question Answering Baselines

We test the relation module on top of our own PyTorch implementation of the BiDAF model (Seo et al., 2017), as well as the recent released BERT base model (Devlin et al., 2018) for the SQuAD 2.0 task. For both of these models, we obtain improvement from adding the relation module. Note that, we do not test our relation module on top of the current leaderboard, as the details are not yet out. We also do not test on top of BERT + Synthetic Self Training (Devlin, 2019) due to lack of computational resources available. We are showing the effectiveness of our method and not trying to compete with the top of the leaderboard.

4.1 BiDAF

We implement the baseline BiDAF model for SQuAD 2.0 task (Clark and Gardner, 2018) with some modifications: adding features that are commonly used in question answering tasks such as TF-IDF, POS/NER tagging, etc, and the auxiliary training losses from (Hu et al., 2019). These modifications to the original BiDAF bring about 3.8% gain of F1 on the SQuAD 2.0 development set (see Table 1).

The input to the relation module is the context vector that is generated from the bi-directional attention flow layer. This context layer is augmented with the hidden states of linear layers trained against plausible answers, which also takes the context layer from the attention flow layer as input. This configuration is shown in Figure 4.

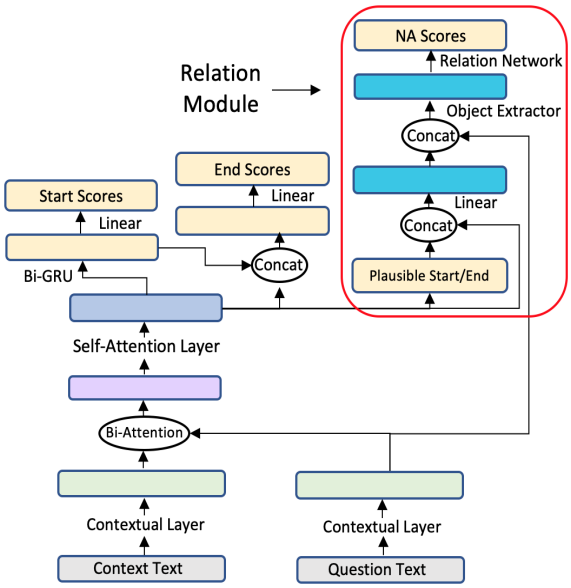


Figure 4: Relation Module applied on BiDAF.

4.2 BERT

BERT is a masked language model pre-trained on large amounts of data that is the core component of all of the current state-of-the-art models on the SQuAD 2.0 task. The input to BERT is the concatenation of a question and context pair in the form of ["CLS"; question; "SEP"; context; "SEP"]. BERT comes with its own special "CLS" token, which is pre-trained on a next sentence pair objective in order to encode entailment information between the two sentences during the pre-training scheme.

We leverage this "CLS" node with the relation module by concatenating it with the output of our Relation Module, and projecting the values down to a single dimension. This combines the information stored in the "CLS" token that has been learned from the pre-training, as well as the information that we learn through our relation module. We allow gradients to be passed through all layers of BERT, and finetune the initialized weights with the SQuAD 2.0 dataset.

5 Experiments

We experiment on the SQuAD 2.0 dataset (Rajpurkar et al., 2018) which contains question and context examples that are crowd-sourced from Wikipedia. Each example contains an answer span in the passage, or an empty string, indicating that an answer doesn't exist. The results are reported on the SQuAD 2.0 development set.

Model	EM(%)	F1(%)
(Clark and Gardner, 2018)	61.9	64.8
Our Implementation of BiDAF	65.7	68.6
BiDAF + Relation Module	**67.7**	**70.4**
BERT-base	73.6	76.6
BERT-base + Relation Module	**74.2**	**77.6**
BERT-large	78.9	82.1
BERT-large + Relation Module	**79.2**	**82.6**

Table 1: Model performance on SQuAD 2.0 development set averaged over three random seeds.

Model	EM(%)	F1(%)
BERT-base	70.7	74.4
BERT-base +Answer Verifier	71.7	75.5
BERT-base + Relation Module	**73.2**	**76.8**

Table 2: SQuAD 2.0 leaderboard numbers on the BERT-base Models. Our model shows improvement over the public BERT-base models on the official evaluation.

We use the following parameters in our BiDAF experiment: 16 context heads, 2 question heads. We set our regularization loss weight for the object extractor to be 0.0005. We use Adam optimizer (Kingma and Ba, 2014), with a start learning rate of 0.0008 and decay the learning rate by 0.5 with a patience of 3 epochs. We add auxiliary losses for plausible answers, and re-rank the non-answerable loss as in (Hu et al., 2019).

BERT comes in two different sizes, a BERT-base model (comprising of roughly 110 million parameters), and a BERT-large model (comprising of roughly 340 million parameters). We use the BERT-base model to run our experiments due to the limited computing resources that training the BERT-large model would take. We only use the BERT-large model to show that we still get improvements with the relation module. The relation module on top of the BERT-base model only contains roughly 10 million parameters.

We use the BERT-base model to run our experiments with the same hyper-parameters given on the official BERT GitHub repository. We use 16 context objects, 2 question heads, and a regularization loss of 0.0005. We also show that on top of the BERT-large model, on the development set, our relation module still obtains performance gain[1]. We use the same number of objects, and the same regularization losses for the BiDAF model experiments.

[1]We do not have enough time to get official SQuAD 2.0 evaluation results for the large BERT models.

	Answerable	Non-Answerable
BERT-base	81.5	78.3
+ Relation Module	**82.1**	**82.1**

Table 3: Prediction accuracies on answerable and non-answerable questions on development set.

Table 1 presents the results of the baseline readers with and without the relation module on the SQuAD development set. Our proposed relation module improves the overall F1 and EM accuracy: 2.0% gain on EM and 1.8% gain on F1 on the BiDAF, as well as 0.8% gain on EM and 1.0% gain on F1 on the BERT-base model. Our relation module is able to take relational information between object-pairs and form a better no-answer prediction than a model without it. The module obtains less gain (0.5% gain of F1) on BERT large model due to the better performance of BERT large model. This module is reader independent and works for any reading comprehension model related to non-answerable tasks.

Table 2 presents performance of three BERT-base models with minimum additions taken from the official SQuAD 2.0 leaderboard. We see that our relation module gives more gain than an Answer Verifier on top of the BERT-base model. Our module gains 1.3% F1 over the Answer Verifier.

Since our relation module is designed to help a MRC model's ability to judge non-answerable questions, we examine the accuracy when a question is answerable and when a question is non-answerable. Table 3 compares these accuracy numbers for these questions with and without the relation module on top of the BERT-base model. The relation module improves prediction accuracy for both types of questions, and with more accuracy gain on the non-answerable questions: close to 4% gain on the non-answerable questions, which is more than 200 non-answerable questions are correctly predicted.

6 Ablation Study

We conduct an ablation study to show how different components of the relation module affects the overall performance for the BERT-base model. First we test only adding plausible answers on top of the BERT-base model, in order to quantify the gain in span prediction that adding these extra answers in would give. We show that with just adding plausible answers, the average of the

Model	EM(%)	F1(%)
BERT-base	73.6	76.6
BERT-base+Plausible Answers	73.5	76.9
BERT-base+RM-Plausible Answers	73.6	76.9
BERT-base+RM (4 heads)	74.1	77.4
BERT-base+RM (16 heads)	**74.2**	**77.6**
BERT-base+RM (64 heads)	74.0	77.2

Table 4: Ablation study on our Relation Module. We experiment with just having plausible answers, just having relation network, and different number of heads for the objects extracted by the relation network. Each of these values are averaged over three random seeds.

three seeds gain only about a 0.3 F1. This gain in F1 is due to the BERT layers being fine-tuned on more answer span data that we provide. Next we study the effects of removing augmenting the context vector with plausible answers. We feed the output of our BERT-base model directly into the object extractor and subsequently to the relation network. This quantifies the effect of forcing the self-attentive heads to focus on a plausible answer span. We notice that this performs comparably to just adding plausible answers, also with only around a 0.3 F1 gain.

Finally, we conduct a study to see the effects of different number of heads on our relation module. We experiment with 4, 16, and 64 heads, with 16 heads performing the best out of these three configurations. Having too few heads hinders the performance due to not enough information being propagated for the relation network to operate on. Having too many heads will introduce redundant information, as well as incorporating extraneous noise for our model to sift through to generate meaningful relations.

7 Analysis

In order to gain better understanding on how the relation module helps on the unanswerable prediction, we examine the objects extracted from the multi-head self-attentive pooling. This is to check whether the relevant semantics are extracted for the relation network. Examples are selected from the development set for data analysis.

In Example 1, the BERT-base model incorrectly outputs "Northridge earthquake" (in red) as the answer. However, after adding our relation module, the model rejects this possible answer and outputs a non-answerable prediction.

The two objects from the question highly attend to token "million" (see the bottom subplot in Figure 5). The top row purple object covers token "1994" , "##ridge earthquake" in the possible answer span window, and "billion" near the end of the context window. We hypothesize that the relation network rejects the possible answer "Northridge earthquake" due to the mismatch of "million" in the question objects and "billion" in the purple context object, and relation scores from all other object pairs.

Example 2 shows another example of non-answerable question and context pair. The BERT-base model incorrectly outputs "input encoding" (in red) as its prediction, while adding our relation module on the BERT-base model predicts correctly that the question is not answerable. Fig-

Example 1
Context: Each year, the southern California area has about 10,000 earthquakes. Nearly all of them are so small that they are not felt. Only several hundred are greater than magnitude 3.0, and only about 1520 are greater than magnitude 4.0. The magnitude 6.7 1994 **Northridge earthquake** was particularly destructive, causing a substantial number of deaths, injuries, and structural collapses. It caused the most property damage of any earthquake in U.S. history, estimated at over $20 billion.
Question: What earthquake caused $20 million in damage?
Answer: None.

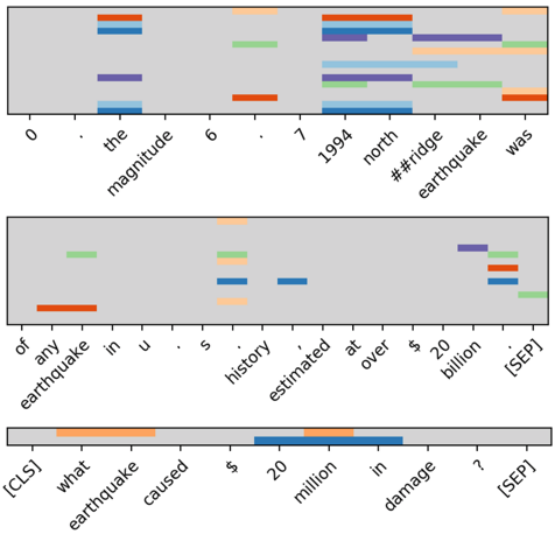


Figure 5: In each subplot, each row represents one object from our object extractor; for each object we highlight the top 5 tokens with highest weights in the entire context and question. We show the two windows where the majority of these top 5 weights occur. For example, the top purple object in the context looks at key phrases such as "##ridge earthquake" in the top subplot and "billion" in the middle subplot; the blue object in the question looks at "20 million in" in the bottom subplot.

Example 2
Context: Even though some proofs of complexity-theoretic theorems regularly assume some concrete choice of **input encoding**, one tries to keep the discussion abstract enough to be independent of the choice of encoding. This can be achieved by ensuring that different representations can be transformed into each other efficiently.
Question: What is the abstract choice typically assumed by most complexity-theoretic theorems?
Answer: None.

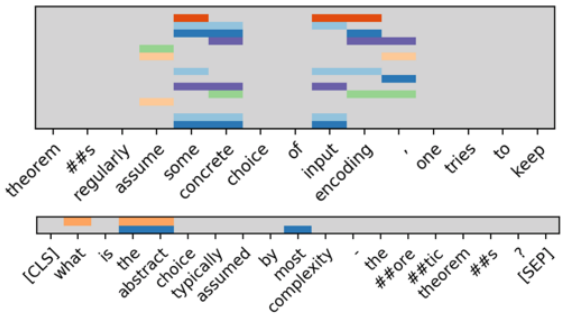


Figure 6: In each subplot, each row represents one object from our object extractor; for each object we highlight the top 5 tokens with highest weights in the entire context and question. We show a window where the majority of the top 5 weights occur. For example, there are numerous objects in the context window that look at the key phrase "some concrete" in the top subplot; the two objects in the question look at the key phrase "the abstract" in the bottom subplot.

ure 6 gives a visual illustration of objects extracted from context and question. In Figure 6, the upper plot illustrates the 16 semantic objects shown in this context window and the lower plot illustrates the two semantic objects from the question. We see that from the upper plot, "some concrete" and "input encoding" are highlighted, while in the lower plot, "what", "the abstract", "most" are highlighted. The mismatch of "the abstract" from the question objects and "some concrete" from the context objects helps indicate that the question is unanswerable.

8 Conclusion

In this work we propose a new relation module that can be applied on any MRC reader and help increase the prediction accuracy on non-answerable questions. We extract high level semantics from multi-head self-attentive pooling. The semantic object pairs are fed into the relation network which makes a guided decision as to whether a question is answerable. In addition we augment the context vector with plausible answers, allowing us to extract objects focused on the proposed answer span, and differentiate from other objects that are not as relevant in the context. Our results on the SQuAD 2.0 dataset using the relation module on both BiDAF and BERT models show improvements from the relation module. These results prove the effectiveness of our relation module.

For future work, we plan to generalize the relation module to other aspects of question answering, including span prediction or multi-hop reasoning.

9 Acknowledgements

We would like to thank Robin Jia and Pranav Rajpurkar for running the SQuAD evaluation on our submitted models.

References

Alec, Karthik Radford, Tim Narsimhan, Salimans, Illya, and Sutskever. 2018. Improving language understanding with generative pre-training.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. *Association for Computational Linguistics*.

Jacob Devlin. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. Lecture Slides.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *Association for Computational Linguistics*, abs/1801.06146.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic reader for machine comprehension. *27th International Joint Conference on Artificial Intelligence*, abs/1705.02798.

Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Ming Zhou. 2019. Read + verify: Machine reading comprehension with unanswerable questions. *Association for the Advancement of Artificial Intelligence*.

Drew A. Hudson and Christopher D. Manning. 2018. Compositional attention networks for machine reasoning. *International Conference on Learning Representations 2018*, abs/1803.03067.

Shoaib Jameel, Zied Bouraoui, and Steven Schockaert. 2018. Unsupervised learning of distributional relation vectors. In *Association of Computational Linguistics*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, abs/1707.07328.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *Conference on Computer Vision and Pattern Recognition*.

Mandar Joshi, Eunsol Choi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. 2018. pair2vec: Compositional word-pair embeddings for cross-sentence inference. *CoRR*, abs/1810.08854.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *Association for Computational Linguistics*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding.

Xiaodong Liu, Wei Li, Yuwei Fang, Aerin Kim, Kevin Duh, and Jianfeng Gao. 2018a. Stochastic answer networks for squad 2.0. *Association for Computational Linguistics*.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018b. Stochastic answer networks for machine reading comprehension. *CoRR*, abs/1712.03556.

Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. 2018c. Stochastic answer networks for machine reading comprehension. In *Association of Computational Linguistics*, pages 1705–1714. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Association of Computational Linguistics 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2018. Coqa: A conversational question answering challenge. *CoRR*, abs/1808.07042.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4967–4976. Curran Associates, Inc.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. *In Proceedings of International Conference on Learning Representations*.

Fu Sun, Linyang Li, Xipeng Qiu, and Yang Liu. 2018. U-net: Machine reading comprehension with unanswerable questions. *Association for the Advancement of Artificial Intelligence*.

Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. 2018. Densely connected attention propagation for reading comprehension. In *Proceedings of Neural Information Processing Systems*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *In Proceedings for Neural Information Processing Systems*, abs/1706.03762.

Wei Wang, Ming Yan, and Chen Wu. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Association of Computational Linguistics*, pages 1705–1714. Association for Computational Linguistics.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S. Yu. 2018. Zero-shot user intent detection via capsule neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. DCN+: mixed objective and deep residual coattention for question answering. *In Proceedings of International Joint Conferences on Artificial Intelligence*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380. Association for Computational Linguistics.

Haoxuan You, Yifan Feng, Xibin Zhao, Changqing Zou, Rongrong Ji, and Yue Gao. 2018. Pvrnet: Point-view relation neural network for 3d shape recognition. *the 33th AAAI Conference on Artificial Intelligence (AAAI2019)*, abs/1812.00333.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *In Proceedings of International Conference on Learning Representations*.

Slot Tagging for Task Oriented Spoken Language Understanding in Human-to-human Conversation Scenarios

Kunho Kim[†], **Rahul Jha**[†], **Kyle Williams**[†], **Alex Marin**[†], **Imed Zitouni**[‡*]
[†]Microsoft Corporation, Redmond, WA, USA
[‡]Google, Mountain View, CA, USA
{kuki, rajh, kywillia, alemari}@microsoft.com, izitouni@google.com

Abstract

Task oriented language understanding (LU) in human-to-machine (H2M) conversations has been extensively studied for personal digital assistants. In this work, we extend the task oriented LU problem to human-to-human (H2H) conversations, focusing on the slot tagging task. Recent advances on LU in H2M conversations have shown accuracy improvements by adding encoded knowledge from different sources. Inspired by this, we explore several variants of a bidirectional LSTM architecture that relies on different knowledge sources, such as Web data, search engine click logs, expert feedback from H2M models, as well as previous utterances in the conversation. We also propose ensemble techniques that aggregate these different knowledge sources into a single model. Experimental evaluation on a four-turn Twitter dataset in the restaurant and music domains shows improvements in the slot tagging F1-score of up to 6.09% compared to existing approaches.

1 Introduction

Spoken Language Understanding (SLU) is the first component in digital assistants geared towards task completion, such as Amazon Alexa or Microsoft Cortana. The input to an SLU component is a natural language utterance from the user and its output is a structured representation that can be used by the downstream dialog components to select the next action. The structured representation used by most standard dialog agents is a semantic frame consisting of domains, intents and slots (Tur and De Mori, 2011). For example, the structured representation of "Find me a cheap Italian restaurant" is the domain **Restaurant**, the intent *find_place*, and slots $[cheap]_{price_range}$, $[Italian]_{cuisine}$, $[restaurant]_{place_type}$. Different sub-tasks within SLU have been extensively studied for human-to-machine (H2M) task completion scenarios (Sarikaya et al., 2016).

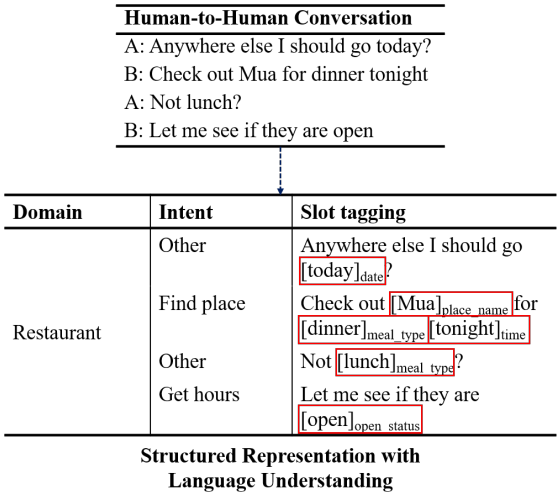


Figure 1: Example of language understanding for task completion on a H2H conversation. In this work, our goal is to identify useful slots (marked with red rectangles).

We extend the task oriented SLU problem to human-to-human (H2H) conversations. A digital assistant can listen to the conversation between two or more humans and provide relevant information or suggest actions based on the structured representation captured with SLU. Figure 1 shows an example of capturing intents and slots expressed implicitly during a conversation between two humans. The digital assistant can show general information about the restaurant Mua, and provide the opening hours based on the captured structured representation. These types of H2H task completion scenarios may allow digital assistants to suggest useful information to users in advance without them needing to explicitly ask questions.

In this paper, we investigate SLU oriented to-

*Work done while the author was at Microsoft Corporation

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 757–767
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

wards task completion for H2H scenarios with a specific focus on solving the **slot tagging** task. Some early conceptual ideas on this problem were presented in DARPA projects on developing cognitive assistants, such as CALO[1] and RADAR[2]. This work can be seen as an effort to formalize the problem and propose a practical framework.

SLU for task completion in H2H conversations is a challenging problem. Firstly, since the problem has not been studied before, there are no existing datasets to use. Therefore, we built a multi-turn dataset for two H2H domains that we found to be prevalent in Twitter conversations: **Music** and **Restaurants**. The dataset is described in more detail in Section 4. Secondly, the task is harder than H2M conversations in several aspects. It is hard to identify the semantics of noisy H2H conversation text with slang and abbreviations, and such conversations have no explicit commands toward the digital assistants requiring the assistant to indirectly infer users intent.

In this work, we introduce a modular architecture with a core bi-directional LSTM network, and additional network components that utilize knowledge from multiple sources including: sentence embeddings to encode semantics and intents of noisy texts with web-data and click logs, H2M based expert feedback, and contextual models relying on previous turns in the conversation. The idea of adding components is inspired from some recent advances in H2M SLU that use additional encoded information (Chen et al., 2016; Su et al., 2018; Kim et al., 2017; Jha et al., 2018). However, these work only considered adding a component from a single knowledge resource. Furthermore, since these additional components bring in information from different perspectives, we also experimented with deep learning based ensemble methods. Our best ensemble method outperforms existing methods by 6.09% for the *Music* domain and 2.62% for the *Restaurant* domain.

In summary, this paper makes the following contributions:

- A practical framework on slot tagging for task oriented SLU on H2H conversations using bidirectional LSTM architecture.
- Extension of the LSTM architecture utilizing knowledge from external sources (e.g. Web data, click logs, H2M expert feedback, and pervious sentences) with deep learning based ensemble methods
- Newly developed dataset for evaluating task oriented LU on H2H conversations

We begin by describing our methods for H2H slot tagging in Section 3. We then describe the data used in our experiments in Section 4 and discuss results in Section 5. This is followed by a review of the related work and conclusion.

2 Related Work

LU involves domain classification, intent classification, and slot tagging (Tur and De Mori, 2011; Sarikaya et al., 2016). Recently various deep neural network (DNN) models have been studied to solve each of these task, such as deep belief network (Sarikaya et al., 2011), deep convex network (Deng et al., 2012), RNN and LSTM (Ravuri and Stolcke, 2015; Mesnil et al., 2015).

Recent advances in LU use additional encoded information to improve DNN based models. There have been some attempts to use data or models from existing domains. One direction is to do transfer learning. Kim et al. (2017) and Jha et al. (2018) utilized previously trained models relevant to the target domain as expert models. They use the output of expert models as additional input to add relevant knowledge while training for the target domain. Goyal et al. (2018) reused low-level features from previously trained models and only retrained high level layers to adapt to a new domain.

There have also been some attempts to use contextual information. Xu and Sarikaya (2014) used past predictions of domains and intents in the previous turn for predicting current utterance. Chen et al. (2016) expanded upon this work by using a set of past utterances utilizing a memory network (Sukhbaatar et al., 2015) with an attention model. Subsequent works attempted to use the order and time information. Bapna et al. (2017) additionally used the chronological order of previous sentences, and Su et al. (2018) used time decaying functions to add temporal information.

Our work trains a sentence embedding that encodes the semantics and intents. DSSM and its variants (Huang et al., 2013; Shen et al., 2014; Palangi et al., 2016) are used for training sentence embedding, which were originally used for finding

[1] https://en.wikipedia.org/wiki/CALO
[2] https://www.cmu.edu/cmnews/extra/030718_darpa.html

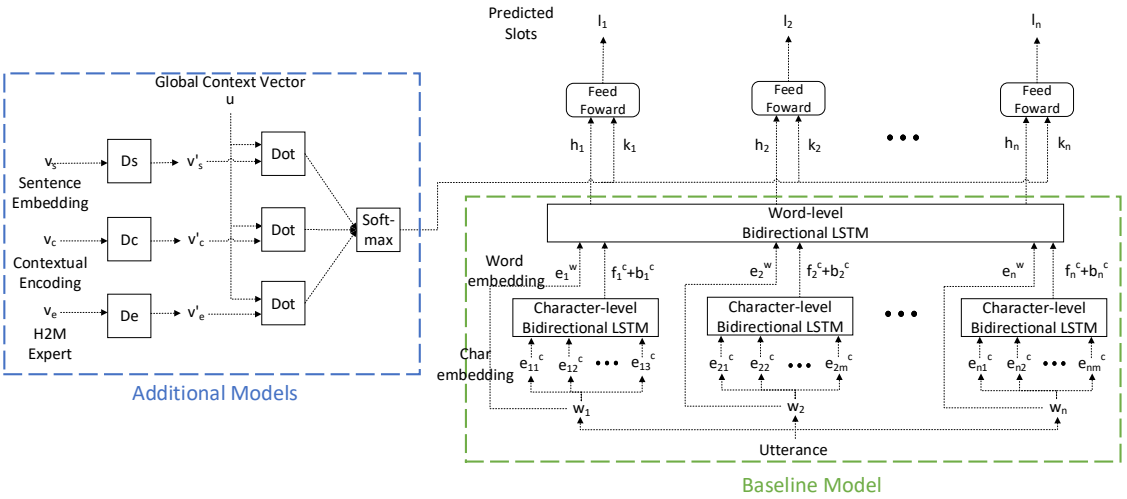


Figure 2: Overview of our slot tagging architecture. Our architecture consists with the core network (Section 3.1) and additional network components utilizing knowledge from multiple sources (Each discussed in Section 3.2.1, 3.2.2, 3.2.3). A network ensembling approach is applied on additional components (Section 3.3), figure shows with the attention mechanism.

relevance between the query and retrieved documents in a search engine. Also there have been attempts to use sentence embeddings similar to our data (Twitter). Dhingra et al. (2016) trained an embedding for predicting hash tags of a tweet using RNNs, Vosoughi et al. (2016) used an encoder-decoder model for sentiment classification.

All of the previous methods have studied LU components for task completion in H2M conversations. On the other hand, prior work on LU on H2H conversations has focused on dialog state detection and tracking for spoken dialog systems. Shi et al. (2017) used CNN model, and later extended multiple channel model for a cross-language scenario (Shi et al., 2016). Jang et al. (2018) used attention mechanism to focus on words with meaningful context, and Su et al. (2018) used a time decay model to incorporate temporal information.

3 Methods

Figure 2 shows the overview of our slot tagging architecture. Our modular architecture is a core LSTM-based network and additional network components that encode knowledge from multiple sources. Slot prediction is done with the final feed forward layer, whose input is the composition of the output of the core network and the additional components. We first describe our core network and then the additional network components, followed by our network ensembling approach.

3.1 Core Network

Our core network is a bidirectional model similar to Lample et al. (2016). The first character-level bidirectional LSTM layer extracts the encoding from a sequence of characters from each word. Each character c is represented with a character embedding $e^c \in \mathbb{R}^{25}$, and the sequence of the embedding is used as the input. The layer outputs

$$f^c = LSTM_{forward}(e^c) \quad (1)$$
$$b^c = LSTM_{backward}(e^c) \quad (2)$$

for each character, where $f^c, b^c \in \mathbb{R}^{25}$.

The second word-level bidirectional LSTM layer extracts the encoding from a sequence of words for each sentence. For each word w_i, the input of the layer is $g_i = f_i^c \oplus b_i^c \oplus e_i^w$ where f_i^c and b_i^c is the output of previous layer, $e_i^w \in \mathbb{R}^{100}$ is the word embedding vector, and $\oplus$ is a concatenation operator of vectors. We use pre-trained GloVe with 2B tweets [3] (Pennington et al., 2014) for the word embedding. The forward and backward word-level LSTM's produce

$$f_i^w = LSTM_{forward}(g_i) \quad (3)$$
$$b_i^w = LSTM_{backward}(g_i) \quad (4)$$

where $f_i^w, b_i^w \in \mathbb{R}^{100}$. Finally, slot l_i is predicted with the last feed forward layer with the input $h_i = f_i^w \oplus b_i^w$.

[3] Downloaded from https://nlp.stanford.edu/projects/glove/

Our model is trained using stochastic gradient descent with Adam optimizer (Kingma and Ba, 2015), with the mini batch size 64 and the learning rate 0.7×10^{-3}. We also apply dropout (Srivastava et al., 2014) on embeddings and other layers to avoid overfitting. The learning rate and dropout ratio were optimized using random search (Bergstra and Bengio, 2012). The core network can be used alone for slot tagging, however we discuss our additional network components in the following sections for improving our architecture.

3.2 Additional Network Components

In this section, we discuss additional network components that encode knowledge from different sources. Encoded vectors are used as additional input to the feed forward layer as shown in Figure 2.

3.2.1 Sentence Embedding for H2H Conversations

Texts from H2H conversations are noisy and contain slang and abbreviations, which can make identifying their semantics challengins. In addition, it can be challenging to infer their intents since there are no explicit commands toward the digital assistants. The upper part of Figure 3 shows part of a conversation from Twitter. The sentence lacks the semantics needed to fully understand "club and country". However, if we follow the URL in the original text, we can get additional information to assist with the understanding. For instance, the figure shows texts found from two sources, **1) web page title of the URL in the tweet** and **2) web search engine queries that lead to the URL in the tweet**. We use web search queries and click logs from a major commercial Web search engines to find queries that lead to clicks on the URL. Using this information, we can infer from the Web page title that the "club and country" referred to in the tweet are Atletico Madrid and Nigeria. Furthermore, the search queries from the search engine logs indicates possible user intents.

In our approach, we encode knowledge found from these two sources based on the URL. In our dataset, we were able to gather 2.35M pairs of tweet text with URL and web search engine queries that lead to the same URL, and 420K pairs of tweet text and web page titles of the URL. We then use this information to train a sentence embedding model that can be used to encode the semantics and implicit intents of each H2H conversation sentence. Our approach is to train a model that projects texts from H2H conversation and texts from each knowledge sources into a same embedding space, keeping the corresponding text pairs close to each other with other non-relevant texts being apart, as shown in Figure 3. The learned embedding model F then can be used to represent any texts from H2H sentences with a vector with semantically similar texts (or similar intents) being projected close to each other in the embedding space. Embeddings are used as additional component of our modular architecture, so that the semantic and intent information can be utilized in our slot tagging model.

We use the deep structured semantic model (DSSM) architecture (Huang et al., 2013) to train the sentence embedding encoder. DSSM uses letter-trigram word hashing, so it is capable of partially matching noisy spoken words so that we can get more robust sentence embeddings for H2H conversations. Let S be the set of sentences from the H2H conversations that have the URL. For each sentence $s \in S$, we find corresponding texts (**web page title of the URL, web search engine queries to the URL**) T_s^+ and randomly choose non-related texts T_s^- from corresponding texts of other sentences (in other words, from different URLs). Like the original DSSM model, each sentence s, $t_s^+ \in T_s^+$, and $t_s^- \in T_s^-$ are initially encoded with letter-trigram word hashing vector x, and used as the input of two consecutive dense layers,

$$x' = W_1 x + b_1 \tag{5}$$

$$y = W_2 x' + b_2 \tag{6}$$

where $x' \in \mathbb{R}^{1000}$ and $y \in \mathbb{R}^{300}$. We train the model to favor choosing $t_s^+ \in T_s^+$ over $t_s^- \in T_s^-$ for each s. So the loss function is defined as minimizing the likelihood,

$$\text{loss} = -\log \prod_{s, T_s^+} P(T_s^+|s) \tag{7}$$

$$P(T_s^+|s) = \prod_{s, t_s^+ \in T_s^+} \frac{\exp(\gamma \text{sim}(s, t_s^+))}{\sum_{t \in T} \exp(\gamma \text{sim}(s, t))} \tag{8}$$

$$\text{sim}(s, t_s^+) = \cos(y_s, y_{t_s^+}) \tag{9}$$

where cos is cosine similarity of two encoded vectors. Please refer to the original paper (Huang et al., 2013) for further details. The dropout ratio,

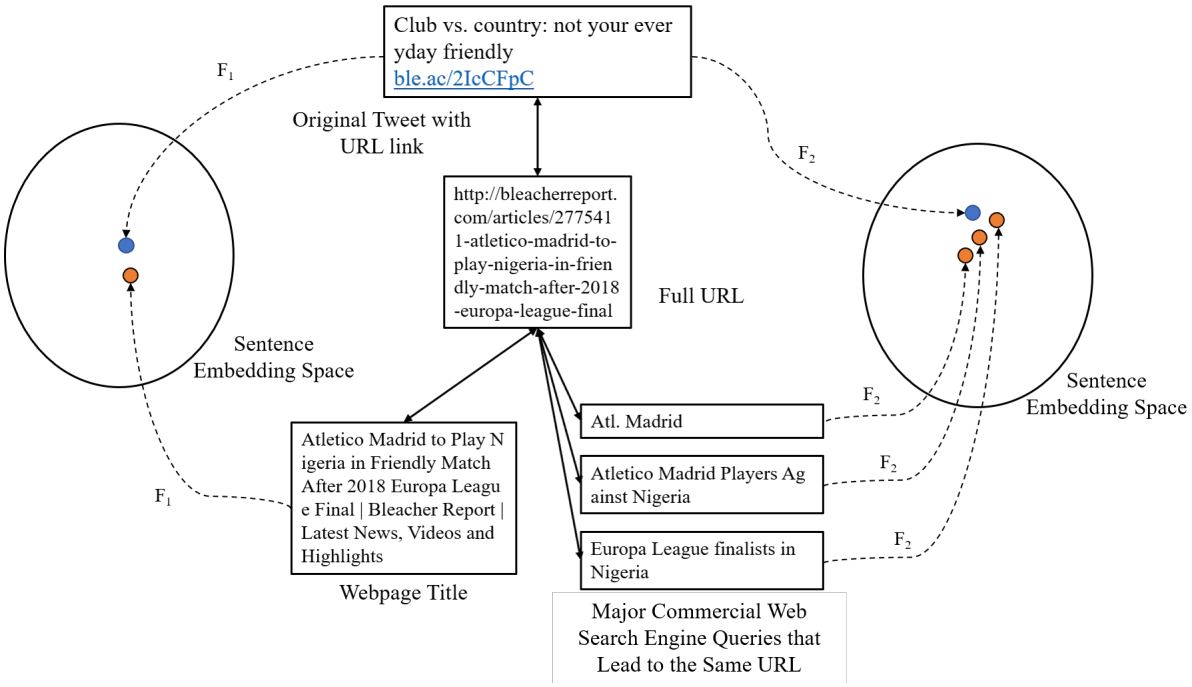


Figure 3: Example of H2H conversation text with URL link and corresponding texts found by following the URL. We use those two sources of corresponding texts to train sentence embedding models. Each model projects the original text and its corresponding texts to a close position in the sentence embedding space, while non-relevant texts are being apart.

learning rate, and γ are selected based on a random search (Bergstra and Bengio, 2012), which are 0.0275, 0.4035×10^{-2}, and 15 respectively. The output of the second dense layer y of trained model is used as the sentence embedding: for each sentence we extract the sentence embedding $v_s \in \mathbb{R}^{300}$.

3.2.2 Contextual Information

Contextual information extracted from previous sentences is known to be useful to improve understanding of human spoken language on other scenarios (Xu and Sarikaya, 2014; Chen et al., 2016; Su et al., 2018). To obtain knowledge from a previous sentence in the conversation, we extract a contextual encoded vector using the memory network (Chen et al., 2016), which uses the weighted sum of the output of word-level bidirectional LSTM h in the core network (Section 3.1) from previous sentences. We did not consider a time decaying model (Su et al., 2018) since our data has a small number of turns.

We tested the model with some variations on 1) number of previous sentences to use and 2) weighting scheme (uniform or with attention). using the implementation from the original paper[4]. From our experiments, the best result was achieved using the previous two sentences with a uniform weight. We use this model to extract the contextual encoded vector $v_c \in \mathbb{R}^{100}$.

3.2.3 Human-to-Machine Expert Feedback

Kim et al. (2017) and Jha et al. (2018) introduced a transfer learning method, which reuses the knowledge from existing trained models on relevant domains (i.e. expert models) to take advantage of previous knowledge to train on a new domain. They extract the output of the expert model and use it as an additional input of feed forward layer for the model on a new domain.

We adopt this idea to take advantage of massive amount of labeled data for H2M conversations. Instead of transferring knowledge from domain to domain, we transfer the knowledge of different tasks within a similar domain. For example, we use **Places (H2M)** domain for the **Restaurant (H2H)** domain, and **Entertainment (H2M)** domain for the **Music (H2H)** domain. We use previously trained slot tagging models on H2M conversations on similar domains as our expert model, which has the same architecture as our core net-

[4] https://github.com/yvchen/ContextualSLU

work (Section 3.1). These H2M models were originally used for the SLU component of a commercial digital assistant. The output of word-level bidirectional LSTM h is then extracted as the encoded vector from H2M expert model $v_e \in \mathbb{R}^{200}$.

3.3 Network Ensemble Approaches

Since additional network components (sentence embedding v_s, contextual information from previous turns of the conversation v_c, and H2M based expert feedback v_e) bring information from different perspectives, we discuss how to compose them into a single vector k with various ensemble approaches.

- **Concatenation**: Here, we simply concatenate all encodings into a single vector,

$$k = v_s \oplus v_c \oplus v_e \tag{10}$$

- **Mean:** We first apply a separate dense layer to each encoded vector to match dimensions and transform into the same latent space, and then take the arithmetic mean of transformed vectors.

$$v'_{\{s,c,e\}} = W_{\{s,c,e\}} + b_{\{s,c,e\}} \tag{11}$$
$$k = \text{mean}(v'_s, v'_c, v'_e) \tag{12}$$

  In the Figure 2, we denote the dense layer applied to each encoded vector $v_{\{s,c,e\}}$ as $D_{\{s,c,e\}}$ for simplicity of representation. Each transformed vector $v'_{\{s,c,e\}} \in \mathbb{R}^{100}$, so $k \in \mathbb{R}^{100}$.

- **Attention:** We apply an attention mechanism to apply different weights on the encoded vectors for each sentence. For our problem, it is not straightforward to define a context vector for each sentence to calculate the importance of each encoded vector; therefore, we adopted the idea of using a global context vector (Yang et al., 2016). The global context vector $u \in \mathbb{R}^{100}$ can be thought as a fixed query of "*finding the informative encoded vector for slot tagging*" used for each sentence. The weight of each encoded vector is calculated with the standard equation of calculating the attention weight, which is the softmax of the dot product of encoding and context vector,

$$w_{\{s,c,e\}} = \frac{\exp(\tanh(v'_{\{s,c,e\}})^\intercal u)}{\sum_{v' \in \{v'_s, v'_c, v'_e\}} \exp(\tanh(v')^\intercal u))} \tag{13}$$

$$k = w_s v'_s + w_c v'_c + w_e v'_e \tag{14}$$

  where $v'_{\{s,c,e\}}$ are same as Equation 11.

The combined single vector k is then aggregated with the output of core network h, $k \oplus h$ is used as the input of the final feed forward layer as shown in Figure 2. The same hyperparameters (mini batch size, learning rate, dropout ratio) and optimizer is used as stated in the baseline model (Section 3.1).

4 Data

Although some datasets with H2H conversations are available (Forsythand and Martell, 2007; Danescu-Niculescu-Mizil and Lee, 2011; Nio et al., 2014; Sordoni et al., 2015; Lowe et al., 2015; Li et al., 2017), they were not feasible to use for experimenting on our task. All datasets excluding the Ubuntu Dialogue (Lowe et al., 2015) were collected without any restrictions on the domain and, as a result, there were insufficient training samples to train a slot tagging model for a specific domain. In addition, the Ubuntu Dialogue dataset (Lowe et al., 2015) focuses on questions related to Ubuntu OS, which is not an attractive domain an intelligent focus that focuses on task completion rather than question answering.

Since there were no existing datasets that were sufficient for our task in H2H conversation, we built our own dataset for the experiments. It was difficult to acquire actual H2H conversations from instant messages due to privacy concerns. Therefore, we chose to use public conversations on Twitter and extracted sequences in which two users engage in a multi-turn conversation. Using this approach, we were able to collect 3.8 million sequences of four-turn conversations using Twitter Firehose.

We focused on two domains for our experiments: **Restaurants** and **Music**. To acquire the dataset for each domain, we first defined a set of key phrases and found the candidate conversations with at least one of those key phrases. Key phrases consisted of the top 100 most frequently used unigrams and bigrams on each relevant domain from the H2M conversation dataset. We used

Restaurant
A: *[lunch]*$_{meal_type}$?
B: *[lunch]*$_{meal_type}$ sounds good. Our routine usually involves sitting at *[Nano's]*$_{place_name}$ with our packed/purchased *[lunches]*$_{meal_type}$ care to join?
A: great. I'll get something from *[physiol]*$_{place_name}$ and meet you there at...?
B: I'll be there in *[5-10 mins]*$_{time}$
Music
A: *[quavo]*$_{media_person}$ got another one
B: I was bout to listen earlier but it said *[feat]*$_{media_role}$ *[Lil Uzi Vert]*$_{media_person}$ lol
A: he *[rap]*$_{media_genre}$ for about two minutes you don't even gotta listen to *[lil uzi]*$_{media_person}$
B: *[quavo]*$_{media_person}$ already a legend man

Table 1: Example conversation in each domain of our dataset

the H2M **Places** domain to find the top n-grams for the **Restaurant** domain and the H2M **Entertainment** domain to find top n-grams for the **Music** domain. **Places** includes other type of places besides restaurants (e.g. tour sights), and also **Entertainment** includes other genre (e.g. movies). So we manually replaced unigrams and bigrams that were not music or entertainment related, and also some terms that are too general (e.g. time, call, find). We were able to gather 16K and 22K candidate conversations for the **Restaurant** and **Music** domains, respectively, using the keyphrases.

We randomly sampled 10K conversations for each domain for annotating slots and domain. Annotation was done by managed judges, who had been trained over time for annotating SLU components such as intents, slots and domains. A guideline document was provided with the precise definition and annotated examples of each of the slots and intents. Agreement between judges and manual inspection of samples for quality assurance was done by a linguist trained for managing annotation tasks. We also ensured that judges did not attempt to guess at the underlying intents and slots, and annotate objectively within the context from the text. We only keep the conversations that are labeled relevant to each domain by annotators. Table 1 shows an example conversation from the dataset in each domain, and Table 2 shows the dataset statistics.

Domain	#Conv.	#Words/Conv.	#Slots
Restaurant	6,514	37.64	15[5]
Music	5,582	44.72	20[6]

Table 2: Statistics of our dataset. Each column shows the number of items in the dataset. "Conv." stands for conversations.

5 Experiments

5.1 Experimental Setup

All experiments were done with 10-fold cross validation for the slot tagging task, and we generated training, development, test datasets using 80%, 10%, and 10% of the data. The development dataset is used for hyperparameter tuning with random search (Bergstra and Bengio, 2012) and early stopping. The baseline is set with core network only (Section 3.1). We evaluated the performance of each of the models with precision, recall, and F1. We checked for statistical significance over the baseline at the p-value < 0.05 using the Wilcoxon signed-rank test.

5.2 Evaluation on Adding Sentence Embeddings for H2H Conversations

In this section, we evaluate adding the sentence embeddings into our slot tagging architecture introduced in Section 3.2.1. Table 3 shows the results of adding sentence embeddings, compared with the baseline and existing sentence embedding methods. We extracted two months of recent tweets that had non-twitter domain URLs in the text for our method. Below is the brief description of each method:

- DSSM (Deep Structured Semantic Model) (Huang et al., 2013): Pre-trained DSSM model from the authors , trained with pairs of *(Major commercial web search engine queries, clicked page titles)*.
- Tweet2Vec (Dhingra et al., 2016): The model was originally used to predict hashtags of a

[5]Slots in **Restaurant** domain include absolute_location, amenities, atmosphere, cuisine, date, distance, meal_type, open_status, place_name, place_type, price_range, product, rating, service_provided, time.

[6]Slots in **Music** domain include app_name, media_award, media_category, media_content_rating, media_genre, media_keyword, media_language, media_lyrics, media_nationality, media_person, media_price, media_release_date, media_role, media_source, media_technical_type, media_title, media_type, media_user_rating, radio_call_sign, radio_frequency

Model	Restaurant			Music		
	P	R	F1	P	R	F1
Core Network (Baseline)	71.23	62.68	66.63	**64.33**	44.14	51.61
+ DSSM (Huang et al., 2013)	70.82	61.60	65.88*	62.83	43.76	51.57
+ Tweet2Vec (Dhingra et al., 2016)	70.58	59.99	64.75*	62.67	41.38	49.79*
+ Ours *(Tweets, Web Search Engine Queries)*	**71.73**	62.38	66.70	63.13	44.09	51.86
+ Ours *(Tweets, Web Paage Titles)*	71.19	**63.51**	**67.11***	63.70	**44.44**	**52.32**

Table 3: Comparison of adding sentence embedding component to our architecture. P, R, F1 stands for precision, recall, F1-score (%) respectively. * denotes the F1-score is statistically significant compared to the baseline.

tweet. We use the pre-trained model from the authors, which used 2M tweets for training.

- Ours *(Tweets, Web Search Engine Queries)*: Trained our model with 2.35M pairs of *(Tweet text with shared URL, Web serach engine queries that lead to the shared URL)*. We extracted most frequent queries (up to eight) found from the major commercial web search engine query logs.
- Ours *(Tweets, Web Page Titles)*: Trained our model with 420K pairs of *(Tweet text with shared URL, web page title of URL)*.

The result shows that adding our proposed sentence embedding network improves the slot tagging result compared to the baseline, while other previous methods have a negative effect. This implies that 1) a sentence embedding specifically trained for H2H conversation texts are needed (compared with original DSSM), 2) our idea of embedding semantics and intentions from web data and search engine query logs can help to improve the slot tagging task (compared to the Tweet2Vec). Since our sentence embedding network trained with web page titles gives the most significant improvement, we used this for further evaluation.

5.3 Evaluation on Utilizing Knowledge Sources

We also tested adding contextual information and H2M expert feedback network components to our slot tagging architecture. Contextual information is extracted from previous two sentences with uniform weighting. For the H2M expert model, we used the pretrained model of **Entertainment** domain for the target **Music** domain, and the **Places** domain for the target **Restaurant** domain for H2M LU.

The upper part (rows 2-4) in Table 4 shows the result of adding each. Results show that 1) adding network component from each knowledge source leads to an improvement on at least one of the domain, 2) improvement on each method varies with the domain. Adding sentence embeddings and contextual information led to significant improvements for the **Restaurant** domain while contextual information and H2M expert feedback led to significant improvements for the **Music** domain.

5.4 Evaulation on Network Ensemble Approaches

We also conducted an experiment to include all network components to see if we can improve further by considering multiple knowledge sources together. The result is shown in the lower part (row 5-7) of Table 4 with different ensembling methods introduced in Section 3.3. It shows that any of the ensemble approaches to add all of the network components leads to better results than adding either of them individually.

The result implies that each of the proposed method improves the slot tagging method from different perspectives so all of them can be considered. Also, we see that attention has the best results among ensemble approaches, with 2.62% higher F1 score for the **Restaurant** domain, and 6.09% for the **Music** domain compared to the baseline. This implies the attention model can help to find the best way to ensemble additional components by predicting the importance of each component for each sentence. Especially, we could see a statistically significant improvement on the **Music** domain compared with other methods. We believe this is because the improvement of each network component on the **Music** domain is more obvious compared to the **Restaurant** domain. We would like to test in other domains for the future work.

Model	Restaurant			Music		
	P	R	F1	P	R	F1
Core Network (Baseline)	71.23	62.68	66.63	64.33	44.14	51.61
+ Sentence Embedding	71.19	63.51	67.11*	63.70	44.44	52.32
+ Contextual	72.74	64.75	68.47*	64.42	49.16	55.72*
+ H2M Expert	71.91	61.21	66.63	64.14	44.67	52.64*
+ Ensemble (Concatenation)	**74.09**	64.89	69.21*	**66.41**	50.10	57.07*
+ Ensemble (Mean)	73.20	**65.43**	69.07*	65.18	51.01	57.11*
+ Ensemble (Attention)	74.03	65.11	**69.25***	66.19	**51.29**	**57.70****

Table 4: Comparison on adding additional network components from each knowledge source and network ensemble approaches that adds all components. P, R, F1 stands for precision, recall, F1-score (%) respectively. * denotes the F1 score is statistically significant compared to the baseline. ** denotes the F1 score of ensemble model is also statistically significant compared to the concatenation ensemble model.

6 Conclusion

We studied slot tagging in H2H online text conversations. Starting from a core network with bidirectional LSTM, we proposed to use additional network components and ensemble them to augment useful knowledge from multiple sources (web data, search engine click logs, H2M expert feedback, and previous utterances). Experiments with our four-turn Twitter dataset on **Restaurant** and **Music** domains showed that our method improves up to 6.09%-points higher F1 on slot tagging compared to existing approaches. For future work, we plan to study our model on domain and intent classification, and also on additional domains.

Acknowledgement

We would like to thank Eric Mei and Soumya Batra for their help on data collection and labeling.

References

Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Sequential dialogue context modeling for spoken language understanding. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 103–114.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Yun-Nung Chen, Dilek Hakkani-Tür, Gökhan Tür, Jianfeng Gao, and Li Deng. 2016. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *INTERSPEECH*, pages 3245–3249.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87. Association for Computational Linguistics.

Li Deng, Gokhan Tur, Xiaodong He, and Dilek Hakkani-Tr. 2012. Use of kernel deep convex networks and end-to-end learning for spoken language understanding. In *IEEE Workshop on Spoken Language Technologies (SLT)*.

Bhuwan Dhingra, Zhong Zhou, Dylan Fitzpatrick, Michael Muehl, and William W Cohen. 2016. Tweet2vec: Character-based distributed representations for social media. In *The 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, page 269.

Eric N Forsythand and Craig H Martell. 2007. Lexical and discourse analysis of online chat dialog. In *International Conference on Semantic Computing (ICSC 2007)*, pages 19–26. IEEE.

Anuj Kumar Goyal, Angeliki Metallinou, and Spyros Matsoukas. 2018. Fast and scalable expansion of natural language understanding functionality for intelligent agents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 3, pages 145–152.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information and knowledge management (CIKM)*, pages 2333–2338.

Youngsoo Jang, Jiyeon Han, Byung-Jun Lee, and Kee-Eung Kim. 2018. Cross-language neural dialog state tracker for large ontologies using hierarchical attention. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2072–2082.

Rahul Jha, Alex Marin, Suvamsh Shivaprasad, and Imed Zitouni. 2018. Bag of experts architectures for model reuse in conversational language understanding. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 3, pages 153–161.

Young-Bum Kim, Karl Stratos, and Dongchan Kim. 2017. Domain attention with an ensemble of experts. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 643–653.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: Amethod for stochastic optimization. In *Proceedings of the 3rd International Conference of Learning Representations (ICLR)*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. Dailydialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 986–995.

Ryan Lowe, Nissan Pow, Iulian V Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 285.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Lasguido Nio, Sakriani Sakti, Graham Neubig, Tomoki Toda, and Satoshi Nakamura. 2014. Conversation dialog corpora from television and movie scripts. In *2014 17th Oriental Chapter of the International Committee for the Co-ordination and Standardization of Speech Databases and Assessment Techniques (COCOSDA)*, pages 1–4. IEEE.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Suman Ravuri and Andreas Stolcke. 2015. Recurrent neural network and lstm models for lexical utterance classification. In *Sixteenth Annual Conference of the International Speech Communication Association*.

Ruhi Sarikaya, Paul A Crook, Alex Marin, Minwoo Jeong, Jean-Philippe Robichaud, Asli Celikyilmaz, Young-Bum Kim, Alexandre Rochette, Omar Zia Khan, Xiaohu Liu, et al. 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 391–397.

Ruhi Sarikaya, Geoffrey E Hinton, and Bhuvana Ramabhadran. 2011. Deep belief nets for natural language call-routing. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5680–5683.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM)*, pages 101–110.

Hongjie Shi, Takashi Ushio, Mitsuru Endo, Katsuyoshi Yamagami, and Noriaki Horii. 2016. A multichannel convolutional neural network for cross-language dialog state tracking. In *IEEE Spoken Language Technology Workshop (SLT)*, pages 559–564.

Hongjie Shi, Takashi Ushio, Mitsuru Endo, Katsuyoshi Yamagami, and Noriaki Horii. 2017. Convolutional neural networks for multi-topic dialog state tracking. In *Dialogues with Social Robots*, pages 451–463. Springer.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Shang-Yu Su, Pei-Chieh Yuan, and Yun-Nung Chen. 2018. How time matters: Learning time-decay attention for contextual spoken language understanding in dialogues. In *Proceedings of the 2018 Conference of the North American Chapter of the Asso-*

ciation for Computational Linguistics: Human Language Technologies (NAACL-HLT), volume 1, pages 2133–2142.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems (NIPS)*, pages 2440–2448.

Gokhan Tur and Renato De Mori. 2011. *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. 2016. Tweet2vec: Learning tweet embeddings using character-level cnn-lstm encoder-decoder. In *Proceedings of the 39th International ACM International conference on Research and Development in Information Retrieval (SIGIR)*, pages 1041–1044.

Puyang Xu and Ruhi Sarikaya. 2014. Contextual domain classification in spoken language understanding systems using recurrent neural network. In *IEEE Internation Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 136–140.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1480–1489.

Window-Based Neural Tagging for Shallow Discourse Argument Labeling

René Knaebel, Manfred Stede
Applied Computational Linguistics
Department of Linguistics
University of Potsdam
Germany
{rknaebel,stede}@uni-potsdam.de

Sebastian Stober
Artificial Intelligence Lab
Faculty of Computer Science
Otto von Guericke University Magdeburg
Germany
stober@ovgu.de

Abstract

This paper describes a novel approach for the task of end-to-end argument labeling in shallow discourse parsing. Our method describes a decomposition of the overall labeling task into subtasks and a general distance-based aggregation procedure. For learning these subtasks, we train a recurrent neural network and gradually replace existing components of our baseline by our model. The model is trained and evaluated on the Penn Discourse Treebank 2 corpus. While it is not as good as knowledge-intensive approaches, it clearly outperforms other models that are also trained without additional linguistic features.

1 Introduction

Shallow discourse parsing (SDP) is a challenging problem in NLP with the aim to identify local coherence relations in text. Discourse relations are used in text to connect individual segments of text logically. The Penn Discourse Treebank (PDTB) (Prasad et al., 2008a) adopts a non-hierarchical view on discourse relations. As an example from the PDTB, the sentence:

- *We would stop index arbitrage* when **the market is under stress**.

contains an explicit discourse relation which is signaled through the underlined connective (`Conn`) and further consists of two arguments (`Arg1` in italics and `Arg2` in bold). In addition, a sense is assigned to a relation, such as `Condition`.

Discourse analysis, however, does not work on a sentence-level, but takes full documents into account. Often, short paragraphs suffice to show the challenge in extracting overlapping relations. To illustrate the problem, we split the paragraphs shown in Figure 1 into small text chunks. Though our implementation works on the level of individual tokens, we here use these chunks to highlight challenges in discourse argument labeling. The chunks are delimited such that no smaller part would play exactly the same role for the various relations involved.

If you think you have stress-related problems on the job, there's good news and bad news. You're probably right, and you aren't alone.

...

Even the courts are beginning to recognize the link between jobs and stress-related disorders in compensation cases, according to a survey by the National Council on Compensation Insurance. But although 56% of the respondents in the study indicated that mental-health problems were fairly pervasive in the workplace, there is still a social stigma associated with people seeking help.

Figure 1: Excerpt from text WSJ 1582 (PDTB corpus)

Text Chunk	R1	R2	R3
If	`Conn`	`Arg1`	
you think you have stress-related problems on the job,	`Arg2`	`Arg1`	
there's good news and bad news.	`Arg1`	`Arg1`	
You're probably right,		`Arg2`	`Arg1`
and		`Arg2`	`Conn`
you aren't alone		`Arg2`	`Arg2`

Table 1: First sample paragraph split into text chunks.

The first example (cf. Table 1) shows that (1) each relations argument contains an arbitrary number of tokens. Further, (2) chunks may have multiple functions (class labels) referring to different individual relations, e.g. the "if" of the first chunk is the connective of **R1**, while in **R2** it is part of the first argument. As a special case, (3) they can have the same class label but pointing to different relations. Finally, (4) there is no generally fixed linear order for the classes `Arg1`, `Arg2`, `Conn`, although

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 768–777
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

Text Chunk	R4	R5
Even the courts are beginning to recognize the link between jobs and stress-related disorders in compensation cases, according to a survey by the National Council on Compensation Insurance.	Arg1	
But	Conn	Arg1
although		Conn
56% of the respondents in the study indicated that mental-health problems were fairly pervasive in the workplace	Arg2	Arg2
there is still a social stigma associated with people seeking help.	Arg2	Arg1

Table 2: Second sample paragraph split into text chunks.

the connective is always syntactically integrated with the second argument.

In the second example (cf. Table 2), (5) arguments can cover whole sentences as demonstrated by **R4**. (6) Although two connectives are next to each other, they can have different arguments and thus constitute different relations. Finally, (7) arguments can also consist of non-continuous chunks as in **R5**.

SDP consists of the main tasks of identifying connectives, demarcating their arguments, assigning senses to them, and finding the senses of so-called *implicit relations* holding between adjacent text spans, which are not explicitly signaled by a connective. Lin et al. (2014) presented a full end-to-end shallow discourse parser that solves these subtasks with a sequential pipeline architecture, which served as a model for the vast majority of follow-up work. Recently, however, most work has addressed specifically the last-mentioned task of identifying the senses of implicit relations, which has been found to be by far the most challenging one.

The focus of our work, in contrast, is on identifying and delimiting the arguments of relations as well as the disambiguation of connectives. Our aim is to do this without any engineering of linguistic features, so that the approach can be easily applied to new corpora and new languages. With this perspective, we follow in particular the proposals of Wang et al. (2015) and Hooda and Kosseim (2017). The first work applies a recurrent neural network on selected sentences and labels these sentences on a token-level. The second work extends this idea and uses an LSTM on a restricted form of the argument labeling task. The authors show the feasibility of a neural model for explicit argument labeling on pre-extracted argument spans. They prepare a dataset of argument spans (extracted from their context) and train a recurrent neural network to label each token's position in such a span .

In our work, we extend this idea to make it applicable within the full SDP setting, i.e., on running text rather than on previously extracted individual relations. As a baseline approach, we use our reimplementation of the system of Lin et al. (2014). We study different applications of our neural model and substitute corresponding components for argument extraction from the baseline pipeline: First we address extracting the arguments of connectives that are already given; this is a sensible assumption since models for connective classification (Pitler and Nenkova, 2009) work quite well. Then, we extend this approach by removing the dependency on previously identified connectives. This step is not easy, because the connectives serve to identify the number of explicit relations in a document. Because the number of relations is initially not clear when connectives are missing, we adapt a sliding window approach for decomposing the text in overlapping windows. We then develop a process of identical *prediction* steps (one for each possible window within a document) and one final *aggregation* step, which combines the individual results into the final set of predicted relations. As an outlook, we study the capacity of our neural model for the joint prediction of explicit and implicit relation arguments.

The main contributions of this paper are

1. integrating a BiLSTM model into the shallow discourse pipeline architecture, and

2. addressing the problem of jointly predicting connective and arguments with a moving-window approach for handling overlapping relations in running text.

In the following, Section 2 discusses relevant related work, and Section 3 explains our method. The experiments and results are presented in Section 4, followed by a discussion in Section 5 and conclusions in Section 6.

2 Related Work

The task of shallow discourse parsing was initiated by the development of the second version

of the Penn Discourse Treebank (PDTB2) (Prasad et al., 2008b) and further by the shared tasks at CoNLL 2015 and 2016 (Xue et al., 2015, 2016). Successful systems at these competitions were those of Wang et al. (2015); Wang and Lan (2016); Oepen et al. (2016). They followed the pipeline model of (Lin et al., 2014), which consists of successive tasks of connective identification, argument labeling, and sense classification for both explicit and implicit relations. Similar to other approaches used for argument extraction (e.g., (Wang et al., 2015; Laali et al., 2016; Oepen et al., 2016)), these competing systems use standard supervised machine learning models in combination with handcrafted linguistic features, such as syntactic, positional, and lexical features.

For argument labeling (or 'extraction'), the exact boundaries of both arguments must be identified. At the CoNLL shared task 2016, Stepanov and Riccardi (2016) reported the best scores for argument extraction with F-measures of 49.64% for `Arg1` and 76.51% for `Arg2`. In contrast to the systems mentioned before, their approach involves conditional random fields (CRF) (Lafferty et al., 2001), which are generally popular for the task of sequence labeling. Also, Ghosh et al. (2011b,a) formulate argument extraction as a token-level sequence labeling problem and use a pipeline of cascaded conditional random fields to mark up each token in a window. On top of a connective classifier, they first predict `Arg2` due to the closeness to the connective. For the prediction of `Arg1`, they use the same feature set as for the former prediction and additionally take the `Arg2` predictions into account. Similar to us, they use a window around the connective (2 sentences before and after the sentence with the connective). They report scores of about 79% F-measure for `Arg2` and 57% F-measure for `Arg1` with their approach. Our approach differs from theirs in that we use tokens instead of full sentences to create windows. This is necessary because a single sentence might contain multiple connectives and participate in more than one discourse relation (cf. the examples shown in Section 1).

A moving-window approach similar to ours is used, for example, by Graves and Schmidhuber (2005) in their work on phoneme classification. The task is quite different for SDP argument labeling, however, as a word's label highly depends on the word's context and the corresponding relation the word is associated with. Thus, we cannot apply such an approach directly and hence define an aggregation procedure for combining the individual per-window predictions.

Argument labeling with recurrent neural networks was done by Wang et al. (2015) in their DCU parser. In addition to word embeddings, they also used hand-crafted features, such as POS tags, syntactic relations, and lexical features. In contrast, our aim is to explore to what extent the problem can be solved *without* feature engineering. Thus, the main inspiration for our approach is the recent work of Hooda and Kosseim (2017), who use an LSTM network on spans of text for labeling arguments. The authors examine their approach on pre-extracted argument spans where the size of spans is determined by the maximal argument pair distance. This shows the feasibility of neural networks to label discourse arguments in a restricted problem setting. Like in our work, they do not rely on additional data other than word embeddings. In contrast to them, we use the embeddings as they are provided, without further adaption throughout the training. In our experiments, we could not identify gains in performance and thus save computation time by reducing trainable parameters.

3 Method

The main goal of our work is to replace existing components in the general discourse parser pipeline framework with our neural model. Depending on which component we want to substitute, we need different methods for processing the discourse.

Our first approach is to replace the argument extraction module, which operates on the basis of previously identified connectives. After that, we introduce an extended model that jointly predicts connectives and their arguments. As demonstrated in Section 1, the main challenge in this task is that a text contains multiple, potentially overlapping, relations that have to be predicted.

Our approach is to decompose the text (and thereby the SDP task) into a series of smaller texts, viz. into a sequence of overlapping windows. For each window, the statistical model is trained to recognize a possibly partial relation. Afterwards, a final aggregation process combines the individual window predictions and thus realizes the argument extraction for a full text.

We describe the baseline discourse parser in

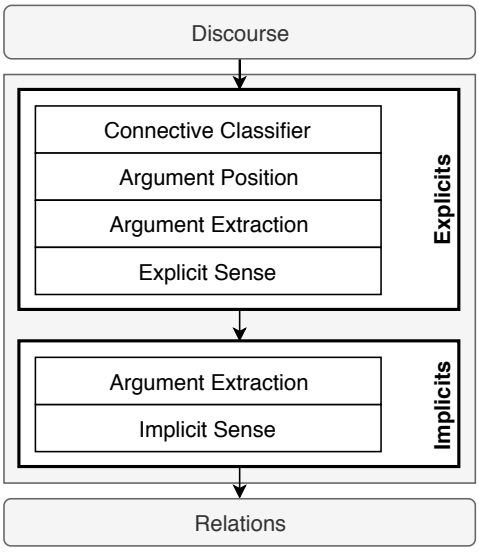


Figure 2: Baseline architecture proposed by Lin et al. (2014) that has been used throughout the experiments.

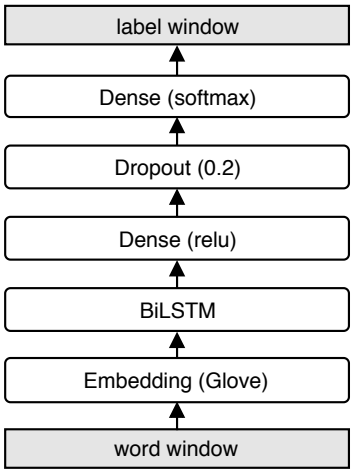


Figure 3: Bidirectional LSTM model architecture. First, tokens in a window are processed sequentially using the recurrent network. Then, each time step is transformed independently using one dense layer for transformation and one dense layer for the final prediction.

Section 3.1. The window model for predicting just arguments is explained in Section 3.2, and its extension to also handle connectives in Section 3.3. Thereafter, we turn to more training and evaluation details in Section 4.

3.1 Baseline: Shallow Discourse Parser

Our baseline discourse parser is inspired by the architecture of the (Lin et al., 2014) parser and consists of several components within a pipeline (see Figure 2). First, explicit relations are identified by classifying connectives, deciding the relative position of the first argument (whether it is contained in the previous sentence or the same sentence as the second argument), then delimiting the arguments, and finally recognizing the sense. In a second phase, implicit relations are classified between adjacent sentences where no explicit relations were found. For the argument extraction component, Lin et al. use a constituent-level approach by iterating over possible subtrees within a sentence and selecting the most likely `Arg1` and `Arg2`.

3.2 Window-based Argument Prediction

For our first approach, we propose a neural network as shown in Figure 3 to predict discourse relations given a sequence of words. Specifically, it operates on a window of words, which we build around a connective that we assume to have been identified by a previous module in the pipeline. Then, for each token the prediction task is defined as a four-way classification problem, i.e., the label is one of `None`, `Arg1`, `Arg2`, or `Conn`[1], as in the work by Hooda and Kosseim (2017) and similar to the window-based approach of Ghosh et al. (2011a).

Each word in the input sequence is embedded into lower-dimensional space. Because of the small size of the PDTB corpus, we use pretrained word embeddings, and these are further processed with a bidirectional Long Short-Term Memory network (BiLSTM). LSTMs have shown better performance compared to simple recurrent neural networks due to their higher capacity for storing important information over longer distances. Further improvements are gained through the bidirectional processing of sequential information (Graves and Schmidhuber, 2005). We keep hidden states for each time step and propagate them to the next layer. For our *BiLSTM* model, the hidden states are processed independently by a dense layer and finally are given to the top output layer (see Figure 3). Between the two dense layers, an additional dropout layer is used for better generalization.

3.3 Joint Prediction of Arguments and Connectives

To apply our model on a text without pre-identified connectives (i.e., to jointly predict connectives and their arguments), we need to adapt training and in-

[1]Notice that while the previously-given connective was used to define the position of the window, we still predict connectives in this model in order to support argument identification (but we will not evaluate the performance on connectives in this model).

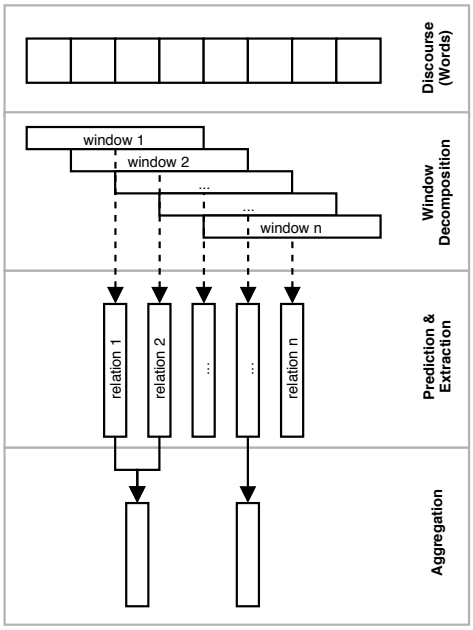


Figure 4: Overview of the proposed method consisting of decomposition, prediction, and aggregation.

ference, as the number of discourse relations and their positions are not known in this scenario. Also, as shown in the introductory example in Table 1, assigning classes to tokens is not a global decision, but has to be decided locally depending on the relation currently predicted. For this reason, we introduce the decomposition of the task into (i) handling multiple overlapping windows and (ii) a subsequent aggregation of the window-level predictions into the final set of predicted relations. The challenge in this aggregation approach is to identify valid predictions and further combine multiple possibly contradictory predictions pointing to the same relation. While we restrict our work here to explicit relations, we will show in our last experiment in Section 4.6 that this approach can also be applied to the case of implicit relations, merely by changing the data that the model is trained on.

3.3.1 Decomposition

The first part of our joint prediction method is to decompose the text into multiple overlapping windows that contain token embeddings. This is done by, first, padding the text at the beginning and at the end such that each token represents the center of one created window.

Then, for each window, we predict whether it contains a (possibly partial) relation or not, and which tokens belong to a particular part of the relation. Thus, for each token our model predicts one of four classes, `None`, `Arg1`, `Arg2`, or `Conn`.

We illustrate the process using the first example from Table 1. We cannot simply split the paragraph into windows and assign classes, because individual tokens have different labels depending on the context. Therefore, we choose the centering of the beginning of `Arg2` as a sufficient criterion (for generality to potentially cover both explicit and implicit relations, we did not choose the connective position as identifier). Thus, we extract three windows around the words "[if][you][think]", "[.][you]['re]", and "[and][you][are]". Because it is possible that a window's center is placed at the beginning or the end of a text, those have to be padded for having the same size as other complete windows.

The main challenge in jointly predicting connectives and their arguments is caused by multiple relations with overlapping arguments. To handle this, we create a unique identifier for each relation by using the position where the second argument of a relation begins. This forces the model to recognize relations only in the case when their second argument is centered in the window, and otherwise ignore them as being not in the focus of the current window.

For training the model, the data is prepared in such a way that it satisfies the properties above. For each relation instance in the PDTB, we use a fixed-size window that is placed on the text such that the relation has `Arg2` centered in this window. To further augment the data, we do not only use perfectly centered relations but additionally move the window one and two words to the left and to the right. Thus, for each relation in a text, we create five training instances. For each window, we keep the words and their argument labels, i.e., whether for a particular relation, a word belongs to one of the arguments, to the connective, or to neither of them. These windows, which correspond to a relation, are collected as positive training samples. Additionally, all windows that have not been extracted so far are gathered as negative training samples. For these negative training samples, although they might identify relations partially, all word labels are set to `None`. In this way, we want to force the model to learn to identify only relations for which `Arg2` is centered within a window.

In short, this process tries to include both arguments of the relation within a single window, but for longer arguments, this is not guaranteed. In this

approach, relations where an argument's span is beyond the window size remain incomplete for the training procedure.

3.3.2 Aggregation

After training a model for individual windows, the remaining challenge is to assemble the final set of relation predictions from all individual window predictions. We can distinguish the following cases: A window can contain

- only `None` labels (and thus no relation is to be extracted at all), or
- an invalid relation, where one or both arguments are empty, or
- a partial relation, where one or both of the arguments is incomplete, or
- a completely predicted relation.

To explain the aggregation step in detail, we define relations more formally and then describe the aggregation of individual relations based on their distance.

A relation is defined as a tuple $\mathrm{R}(a_1, a_2, c)$ of three sets, one for the first argument, the second argument, and the connective. Each of these sets contains the position indices of the included words, and hence these three sets are, by definition, disjoint. We maintain this property throughout the aggregation process of two relations.

Given two relations r_1, r_2, we define the merge of two relations, $r = r_1 \cup r_2$, as follows:

$$r^c = (r_1^c \cup r_2^c) \tag{1}$$

$$r^{a_2} = r_1^{a_2} \cup r_2^{a_2} \setminus r^c \tag{2}$$

$$r^{a_1} = (r_1^{a_1} \cup r_2^{a_1}) \setminus (r^c \cup r^{a_2}) \tag{3}$$

The crucial step is to decide whether two relations that are labeled in two consecutive windows are to be merged, i.e., to decide whether they denote the same relation. To this end, we measure the *distance* between two relations and merge them if the distance is below a certain threshold. Here we make use of the Jaccard distance as a distance metric for sets (but other metrics could be used as well). The Jaccard distance is based on the Jaccard index, which measures the similarity between finite sets by comparing the union and intersection of those sets:

$$J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

	None	Arg1	Arg2	Conn
Exp1	70.68 %	14.43 %	13.79 %	1.10 %
Exp2	85.34 %	7.21 %	6.89 %	0.56 %
Exp3	82.78 %	8.60 %	8.37 %	0.25 %

Table 3: Class label ratio calculated on the extracted training windows.

We define the distance of two relations $J_{rel}(r_1, r_2)$ as the mean of the two Jaccard distances between the relation's arguments.

$$J_{rel}(r_1, r_2) = \frac{J(r_1^{a_1}, r_2^{a_1}) + J(r_1^{a_2}, r_2^{a_2})}{2}$$

Note that for the calculation of the distances of the second arguments, we regard the connective as part of second arguments. This gives some, but not too much, influence to the connectives on the distance measure, compared to the influence of the arguments.

4 Experiments and Results

As described above, we experiment with two tasks of different complexity. The first experiment is a simplification of the general argument labeling problem, where a connective has already been classified. Because the position of the window is thus already determined, no aggregation is necessary for this scenario. For the second and third experiment, the exact positions of relations are not given and thus we follow our sliding-window approach. In all our evaluations, we use precision, recall, and F1 score of exact matches, in order to be comparable with the previous work.

All models are trained for 25 epochs on the corresponding training set, specific for a certain task. We use pretrained word embeddings with 300 dimensions as described later in Section 4.1. Each LSTM has a hidden layer of size 512. The output of each LSTM is concatenated per step, thus, the output of the BiLSTM results in 1024 dimensions per step. The dense layer with ReLU activation function on top of the BiLSTM has 64 dimensions, and the dropout works with a 0.2 probability. Finally, all models are trained using the Adam optimizer (Kingma and Ba, 2015). Because the classes are unbalanced throughout the experiments (compare Table 3) the cross-entropy loss is weighted (King and Zeng, 2001) according to the per-class occurrences in the extracted training windows.

4.1 Word Embedding

After the extraction of a vocabulary from the training corpus, we identify each token by a unique index. Every word that is not listed in the vocabulary is substituted by a special index for unknown words.

Because of the small size of the corpus, it is common to use pretrained embeddings (Mikolov et al., 2013) instead of training them solely on the given training corpus. In our experiments, we use *global vectors* for word representations (GloVe) (Pennington et al., 2014), similar as done by Hooda and Kosseim (2017). In general, GloVe embeddings have yielded promising results for a variety of related tasks. Hence, in the present study, we use GloVe without comparatively evaluating different pretrained embeddings.

Since both models, the word embedding model and ours, are trained on different corpora, the vocabularies also differ. First, we initialize all words with a random embedding. Then, for each word that is found in the set of pretrained embeddings, we replace the random embedding by its pretrained equivalent.

A minor disadvantage of word embeddings is that they are learned on a syntactic level. This means that although two words look similar in terms of characters, they might have different meanings depending on the words in their context. As a consequence, we follow the idea of using LSTMs on top of the word embeddings in order to account for the context in the individual representation.

In contrast to Hooda and Kosseim (2017), we avoid dynamically retraining the word vectors throughout the training process. Our earlier experiments showed that in our setting, training the embeddings has no positive effect on the final result, and thus, by avoiding this step, we also reduce the number of parameters that have to be trained.

4.2 Data Preparation

In our experiments, we follow the CoNLL shared task on shallow discourse parsing and work on their prepared dataset. This dataset differs slightly from the original PDTB2 corpus which is caused by the merge of a few sense labels and the cleaning of the PDTB data. For each relation in the CoNLL corpus, we extract a fixed-size window where the second argument is centered, i.e., where `Arg2` starts in the middle of the window. To produce more training data, we also extract windows that are shifted by small margins (up to two positions) to the left and to the right. Each of these windows constitutes a positive example of a discourse relation the model should learn. Conversely, every window of the same size where `Arg2` is not centered is added as a negative example. Even though a negative example may contain parts of some arguments, the words are labeled with `None` in order to force the model to only recognize relations where `Arg2` starts in the middle of the window. We keep punctuation as part of the vocabulary, as it might capture relevant discourse information.

In our evaluations, all models are trained on a window size of 100 tokens. Based on the span lengths found in the corpus, shown in Table 4, we choose this window size, as it captures a solid majority (over 80%) of the relations.

	Explicit		Implicit	
Perc.	**Span Length**	**Distance**	**Span Length**	**Distance**
min	2	0	2	0
20 %	14	2	20	2
40 %	21	2	29	2
60 %	29	3	38	2
80 %	44	4	49	3
max	1167	987	437	249

Table 4: Statistics calculated from the CoNLL2016 dataset. Overview of span lengths containing both arguments and distances of arguments for specific percentiles.

	Precision	**Recall**	**F1**
		Explicits	
Conn	83.42	79.82	81.58
Arg1	28.75	27.51	28.12
Arg2	45.54	43.57	44.54
Arg1+Arg2	27.70	26.51	27.09
		Non-Explicits	
Arg1	67.85	36.05	47.08
Arg2	67.65	35.94	46.94
Arg1+Arg2	59.94	31.84	41.59
		All	
Conn	83.42	79.82	81.58
Arg1	51.32	34.89	41.54
Arg2	58.28	39.62	47.17
Arg1+Arg2	45.38	30.86	36.74

Table 5: Evaluation of our baseline pipeline architecture. [2]

4.3 Baseline

The baseline model is inspired by Lin et al. (2014) who proposed a pipeline approach of several components. For our comparison, we focus on those

Label	Precision	Recall	F1
Arg1	46.69	44.59	45.62
Arg2	68.94	65.83	67.35
Arg1+Arg2	48.16	45.99	47.05

Table 6: Exact match of explicit arguments for a given connective.

components that are responsible for argument extraction: the connective classifier, argument position classifier and argument extractor in the explicit case, as well as the extraction of non-explicit arguments (cf. Figure 2). In contrast to our own model, which works on token level prediction, Lin et al.'s argument extraction model works on subtree level.

The performance reported in Table 5 shows the evaluation scores following the CoNLL shared task for our reimplementation of the pipeline described by Lin et al. Although these numbers are lower than those reported for the original implementation (e.g. predicting both explicit arguments is 10% worse), they may serve as a comparison and indicate that a re-implemenation of this system is not trivial. The values give different perspectives on the data, a full evaluation (**All**), and more detailed views on both parts (**Explicits** and **Non-Explicits**), which correlate with the structure of the architecture.

4.4 Connective Arguments

In the first experiment, we train a model to substitute the components for argument position and argument extraction similar to Ghosh et al. (2011a). The training data contains only positive explicit samples, since the model is never applied to other situations than these. With our procedure described above, we extract 73.610 samples from the corpus. For inference, we use a fixed-size window centered around the previously identified connective. The labeled indices for `Arg1` and `Arg2` are taken without any further processing.

The results in Table 6 show an increase of the values for `Arg1` and `Arg2` compared to our baseline.

4.5 Explicit Argument Extraction

The second experiment generalizes the window-based model and predicts arbitrary explicit relations for a text. This is challenging compared to the former experiment, because the connective is missing and therefore the model does not know the exact position of a relation. For this reason, we introduced the decomposition of a discourse with the sliding window approach and further introduce an aggregation method to get the final set of predicted relations.

Label	Precision	Recall	F1
Conn	69.25	44.56	54.23
Arg1	36.52	23.50	28.59
Arg2	62.96	40.51	49.30
Arg1+Arg2	40.29	25.93	31.55

Table 7: Exact match of explicit arguments for joint prediction of connectives and their arguments.

	Precision	Recall	F1
	Explicits		
Conn	71.35	62.73	66.76
Arg1	33.16	29.15	31.03
Arg2	52.47	46.13	49.09
Arg1+Arg2	37.25	32.75	34.86
	Non-Explicits		
Arg1	41.99	29.26	34.49
Arg2	44.32	30.88	36.40
Arg1+Arg2	40.16	27.99	32.99
	All		
Conn	71.35	62.73	66.76
Arg1	40.95	31.77	35.78
Arg2	51.47	39.94	44.98
Arg1+Arg2	41.83	32.45	36.55

Table 8: Evaluation of the joint extraction of explicit and non-explicit arguments.

The training data additionally contains negative explicit samples in contrast to the first experiment. The same amount of negative instances as positive instances is sampled from all possible negative instances. We extract twice as much training instances as before.

As shown in Table 7, the scores for connective identification are not as high as achieved with a specialized model (as in the baseline). Further, the sores for argument extraction also decrease for `Arg2` slightly and for `Arg1` even more. This is probably caused by the unbalanced data, as `None` labels occur much more often than other labels (see Table 3).

4.6 Explicit/Implicit Arguments Extraction

In the final experiment, we examine the full capacity of our model by using it for jointly predicting both explicit and implicit relation arguments. The training data consists of explicit and implicit relation instances, and for each positive sample, one negative sample is added to the training data. In sum, the model is trained on 325.350 instances.

As expected, the number for identifying non-

explicit relations is lower than the baseline (see Table 8). In contrast to our model, the baseline always selects two adjacent unlabeled sentences, which achieves good results despite its simplicity. Our models seems unable to recognize any sensible underlying pattern.

5 Discussion

Usually, different specialized models are combined to label arguments, either on a constituent-level (Lin et al., 2014; Kong et al., 2014) or on a token-level (Ghosh et al., 2011a,b). As we explained earlier, though, a crucial difference is their usage of highly-engineered linguistic features (e.g., cue words, syntactic classes, word pair relations, production rules), which our system does not use. Instead, it relies solely on word (token) sequences encoded by pretrained embeddings, which thus represent a certain amount of semantic information.

Our comparison with the re-implemented pipeline architecture is quite weak, as our implementation does not perform as good as the original work that relies on manually-engineered rules.

While our approach can obviously not compete with the knowledge-intensive approaches of Oepen et al. (2016) and Stepanov and Riccardi (2016), a fair comparison is that to the similar approach of Wang et al. (2015); here, our system clearly outperforms the earlier result. They report exact match F-measure of 36.32% for `Arg1` and 41.70% for `Arg2`. Compared to that system, our first experiment's approach where the connective was given (Section 4.4) performs much better. In comparison with our second experiment, our system's scores are lower for `Conn` and `Arg1`, but still higher for `Arg2` and both arguments extraction. A big problem in joint connective–argument extraction is the limited amount of data and the unbalanced class labels. We tried to work on the second problem by using weighted losses based on the class occurrences.

6 Conclusions and Future Work

In this work, we integrate a BiLSTM model in the shallow discourse parsing framework. We described tasks of different complexity in argument labeling and explained different ways of applying our neural model. For the general task of argument labeling, we adapt a token-level window-based approach and introduce a novel aggregation method, which is needed for combining individual predictions into the final set of relation arguments. We explored the limits of this approach by studying the joint prediction of the arguments of explicit and implicit relations.

The aggregation of partial relations is done using a distance threshold. For gaining possible improvements on the final results, it may well be worth studying different methods for merging conflicting relation predictions as well as defining different ways of computing the distances of individual predictions.

Because of the simplified nature of our architecture, we think that there are further interesting potentials for future work. Formulating the new window-training problem makes it possible to easily replace our proposed model by other architectures with more capacity. At the same time, we see this way of solving argument extraction as one step toward reducing the complexity and degree of error propagation in the more traditional SDP pipeline architecture.

We release the source code of our system[3] to promote future research.

Acknowledgments

This research was supported by the Federal Ministry of Education and Research (BMBF), Contract 01IS17059.

References

Sucheta Ghosh, Richard Johansson, Giuseppe Riccardi, and Sara Tonelli. 2011a. Shallow discourse parsing with conditional random fields. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1071–1079. Asian Federation of Natural Language Processing.

Sucheta Ghosh, Sara Tonelli, Giuseppe Riccardi, and Richard Johansson. 2011b. End-to-end discourse parser evaluation. In *Fifth IEEE International Conference on Semantic Computing (ICSC), 2011; September 18-21, 2011; Palo Alto, United States*, pages 169–172.

A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052 vol. 4.

Sohail Hooda and Leila Kosseim. 2017. Argument labeling of explicit discourse relations using lstm

[3] www.github.com/rknaebel/discopy

neural networks. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 309–315. INCOMA Ltd.

Gary King and Langche Zeng. 2001. Logistic regression in rare events data. *Political Analysis*, 9:137–163.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Fang Kong, Hwee Tou Ng, and Guodong Zhou. 2014. A constituent-based approach to argument labeling with joint inference in discourse parsing. In *EMNLP*.

Majid Laali, Andre Cianflone, and Leila Kosseim. 2016. The clac discourse parser at conll-2016. In *Proceedings of the CoNLL-16 shared task*, pages 92–99. Association for Computational Linguistics.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20:151–184.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Stephan Oepen, Jonathon Read, Tatjana Scheffler, Uladzimir Sidarenka, Manfred Stede, Erik Velldal, and Lilja Øvrelid. 2016. Opt: Oslo–potsdam–teesside. pipelining rules, rankers, and classifier ensembles for shallow discourse parsing. In *Proceedings of the CoNLL-16 shared task*, pages 20–26. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics.

Emily Pitler and Ani Nenkova. 2009. Using syntax to disambiguate explicit discourse connectives in text. In *ACL/IJCNLP*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008a. The penn discourse treebank 2.0. In *LREC 2008*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008b. The penn discourse treebank 2.0. In *In Proceedings of LREC*.

Evgeny Stepanov and Giuseppe Riccardi. 2016. Unitn end-to-end discourse parser for conll 2016 shared task. In *Proceedings of the CoNLL-16 shared task*, pages 85–91. Association for Computational Linguistics.

Jianxiang Wang and Man Lan. 2016. Two end-to-end shallow discourse parsers for english and chinese in conll-2016 shared task. In *Proceedings of the CoNLL-16 shared task*, pages 33–40. Association for Computational Linguistics.

Longyue Wang, Chris Hokamp, Tsuyoshi Okita, Xiaojun Zhang, and Qun Liu. 2015. The dcu discourse parser for connective, argument identification and explicit sense classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 89–94. Association for Computational Linguistics.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 1–16. Association for Computational Linguistics.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Attapol Rutherford, Bonnie Webber, Chuan Wang, and Hongmin Wang. 2016. Conll 2016 shared task on multilingual shallow discourse parsing. In *Proceedings of the CoNLL-16 shared task*, pages 1–19. Association for Computational Linguistics.

TILM: a Neural Language Model with Evolving Topical Influence

Shubhra Kanti Karmaker Santu
and **Kalyan Veeramachaneni**
MIT LIDS
Cambridge, Massachusetts, USA
santu@mit.edu
kalyanv@mit.edu

ChengXiang Zhai
Department of Computer Science
University of Illinois Urbana Champaign
Urbana, Illinois, USA
czhai@Illinois.edu

Abstract

Content of text data are often influenced by contextual factors which often evolve over time (e.g., content of social media are often influenced by topics covered in the major news streams). Existing language models do not consider the influence of such related evolving topics, and thus are not optimal. In this paper, we propose to incorporate such topical-influence into a language model to both improve its accuracy and enable cross-stream analysis of topical influences. Specifically, we propose a novel language model called Topical Influence Language Model (TILM), which is a novel extension of a neural language model to capture the influences on the contents in one text stream by the evolving topics in another related (or possibly same) text stream. Experimental results on six different text stream data comprised of conference paper titles show that the incorporation of evolving topical influence into a language model is beneficial and TILM outperforms multiple baselines in a challenging task of text forecasting. In addition to serving as a language model, TILM further enables interesting analysis of topical influence among multiple text streams.

1 Introduction and Motivation

Language modeling plays a central role in many Natural Language Processing (NLP) tasks and applications. Neural language models have attracted much attention recently due to their superior performance (Dieng et al., 2016; Kiros et al., 2014; Kiddon et al., 2016; Ranzato et al., 2015; Devlin et al., 2018; Peters et al., 2018). One common limitation of the existing neural language models is that they cannot model the potential influence of related contextual factors on text content generation. However, as text data are produced by humans based on their observations of the real world, the content of text data are generally influenced by many contextual factors, and thus, it is necessary to model the influence of those contextual factors on the generation of text content in order to optimize language modeling. For example, the content of social media may be influenced by the popular topics in news stream; another example is that the content of research papers in one research community such as *Information Retrieval* are often influenced by the topics of research papers published in the same community in the past or research papers published in another related research community such as *Machine Learning* since the general algorithms developed in the latter may be applied to solve application problems in the former.

In this paper, we propose to incorporate such topical influence into a language model to both improve its accuracy and enable cross-stream analysis of topical influences. Specifically, we propose a novel language model called Topical Influence Language Model (TILM), which is a novel extension of a neural language model to capture the influence on the contents in one text stream by the evolving topics in another related (or possibly same) text stream. In other words, TILM is a recurrent neural network-based deep learning architecture that incorporates topical influence to model the generation of a dynamically evolving text stream. Since most text data have time stamps associated with them, which generally indicate the time when a text document was produced, text data can often be regarded as a sequence / stream of text objects ordered by their time stamps. TILM is designed to model the generation of such a text stream, i.e., the generation of text data conditioned on a given time stamp. We also assume that the distributions of words in the text data from two different timestamps are somewhat different, which allows us to capture the evolution of topics in the stream data.

TILM is comprised of three basic components. The first component is the *Sequence Generator*, which can be any current neural language model.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 778–788
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

The second component is the *Topic Generator* which captures the trend of different topics of interest within the *influencing text stream*, which can be based on any topic models, e.g., LDA. Finally, the third component is a novel *Influence Generator*, which ensures that TILM can capture topical influence corresponding to the input timestamp and incorporate it into the generation process to ensure that the generated text is consistent with that particular timestamp from the perspective of topical influence. TILM combines these three essential components into a single unified model and learns all their optimal parameter values from a training text stream corpus with time stamps.

We conducted comprehensive experiments with six different sets of publication stream datasets to evaluate the effectiveness of TILM. These datasets contain titles of the papers published in different machine learning theory and applied machine learning conferences over 20 years, which were collected from the Open Academic Graph. Experimental results show that the incorporation of topical influence into a language model is beneficial and TILM outperforms multiple baselines by a clear margin in a challenging task of text forecasting. We found that effectively capturing the evolving topical influence is the key to improving the accuracy of language modeling. In addition to serving as a language model, TILM further enables interesting analysis of topical influence among multiple text streams.

2 Related Works

There has been a surge of research interest in the use of neural network (NN) architectures for language modeling and automatic text generation in recent years (Dieng et al., 2016; Kiros et al., 2014; Kiddon et al., 2016; Ranzato et al., 2015). The first NN-based text generator was proposed by Kukich (Kukich, 1987), although generation was done only at the phrase level. Recent advances in recurrent neural network-based language models (RNN-LM) have demonstrated the value of distributed representations and its power to model arbitrarily long dependencies (Mikolov et al., 2010, 2011). Sutskever et al. (Sutskever et al., 2011) introduced a simple variant of the RNN that can generate meaningful sentences by learning from a character-level corpus. Mao et.al. have demonstrated how Recurrent Neural Networks, specially, Long-Short-Term-Memory (LSTM) is effective in solving various text generation tasks (Mao et al., 2014). TopicRNN proposed by Dieng (Dieng et al., 2016) integrated the merits of RNNs and latent topic models to capture long-range semantic dependency. Recently, Generative Adversarial Nets (GANs) that use a discriminative model to guide the training of the generative model has shown promising results in automated text generation (Rajeswar et al., 2017; Lin et al., 2017; Che et al., 2017; Zhang et al., 2017). All these text generation techniques have also been vastly studied for generating summary for text corpora [see (Gambhir and Gupta, 2017) for a comprehensive survey]. Most recently, BERT (Devlin et al., 2018) and ELMo (Peters et al., 2018) have shown very promising performance. However, none of these existing methods can model topical influences from related contextual factors on generation of text data. While external topical influences have been studied in the context of search behavior modeling (Karmaker Santu et al., 2017, 2018), similar study has not been pursued yet for text generation modeling. Specifically, current text generation processes are static processes with no notion of time and thus, can not model dynamically evolving text stream data corresponding to evolving influences of related text streams, which the proposed TILM can do.

3 Topical Influence Language Model

We first briefly discuss some preliminaries and notations. Next, we present the three major components that are the building blocks of the proposed Topical Influence Language Model (TILM) and then present how TILM combines them into a single unified model.

3.1 Preliminaries and Notations

The goal task involves performing language modeling on each text stream within a set of text streams, $S = \{s_1, s_2, ..., s_m\}$, where, $|S| = m$. Each text stream consists of a set of tuples in the form of (document, timestamp) , e.g., $s_1 = \{(d_1, t_1), (d_2, t_2), ...\}$. Each document is a sequence of words, e.g., $d_1 = [x_1 x_2 ...]$, where, each $x_i \in V$ and V is the vocabulary set. For modeling purposes, we group the documents within each stream into different bins based on their corresponding timestamp. The time ranges $[t_{start}, t_{end}]$ used to create these bins are defined by the user and can be in varying granularity like year, month,

day or seconds. Each bin is then assigned a discrete timestamp T and all the documents in that bin share the same timestamp, e.g., all paper published in different machine learning conferences during year 2016 are assigned the timestamp $T_{2016} = [Jan_{2016}, Dec_{2016}]$.

3.2 Sequence Generator

Sequence Generator is the basic component of TILM which generates the next word x_i in the sentence given $i-1$ previous words. Thus, *Sequence Generator* is essentially a language model which generates a probability distribution over a sequence of words that can be used to predict the next word in the sequence. Any sequence modeling framework, e.g., Hidden Markov Models, Recurrent Neural Networks etc. can work as a sequence generator. For the experiments described in this paper, we chose recurrent neural network with LSTM cells as the *Sequence Generator* due to its recent promising results obtained for language modeling tasks (Kiros et al., 2014; Kim et al., 2016). Given the previous i words, i.e., $x_{1:i}$, the recurrent neural network based language models compute the conditional probability for the next word $y_i = v$ for $v \in V$, the vocabulary set, by computing a hidden state h_i and passing it through a Softmax function:

$$P(y_i = v|x_{1:i}) \equiv P(y_i = v|h_i) \quad (1)$$

$$P(y_i|h_i) \propto \exp(W^{\Omega}h_i + B^{\Omega}) \quad (2)$$

$$h_i = \Omega(h_{i-1}, x_i) \quad (3)$$

Here, Ω can be a standard RNN cell or more complicated cell like LSTM, GRU etc and W and B are linear transformation coefficients. Output at step i, i.e., y_i is fed as input for step $i+1$, thus, $x_{i+1} = y_i$.

3.3 Topic Generator

The next component of TILM is the *Topic Generator*. The primary purpose of this component is to analyze different topics across a related text stream data (let us call it s) and compute the evolution of topic distributions within that stream over time. It takes all past text stream data of s as input and applies a probabilistic topic model to infer n (a user defined parameter) different topics, each represented with a unique distribution over the entire vocabulary. Again, many different topic models can be potentially used, but in our experiments, we chose LDA (Blei et al., 2003) as the *Topic Generator* since it has been the most popular topic modeling technique in the last decade. Once n topics are identified, the *Topic Generator* takes all text content from each discrete timestamp T (defined in section 3.1) separately and computes the distribution of n topics over each timestamp, which we denote by θ_T.

The *Topic Generator* also provides a subcomponent, i.e., *History Extractor*, which, given a particular timestamp T as input, retrieves the topic distributions of previous r (a user defined parameter) timestamps computed by LDA. We mathematically denote the output of *History Extractor* by $\theta_{T-r:T-1}$, where, $\theta_{i:j}$ denotes topic distributions from timestamp i to j augmented into a single vector. This means the cardinality of vector $\theta_{T-r:T-1}$ is $r \times n$. In the case of modeling influence from multiple related text streams and assuming we have m such streams, we can simply concatenate $\theta_{T-r:T-1}$ from each stream to create a single vector of dimension $r \times n \times m$.

3.4 Influence Generator

The *Influence Generator* is a pivotal component of TILM, which models the evolving topical influence during the text generation process. Given a particular timestamp T, we represent topical influence by a real valued vector (γ_T) of dimension K (another user defined parameter), which is essentially the output of *Influence Generator*. The input to the *Influence Generator* is the $r \times n$ dimensional vector of topic distributions from previous r timestamps of a related text stream, i.e., $\theta_{T-r:T-1}$ (assuming T as the current timestamp). Thus, *Influence Generator* essentially maps a $r \times n$ dimensional topic vector to a K dimensional influence vector. Although any function that can perform this mapping can resemble as Influence Generator, we chose a feed-forward neural network for TILM due to its capability of approximating a wide family of functions. Without loss of generality, we used ReLU activation units in the hidden layers. Once the influence vector (γ_T) is computed, it is then injected as a bias into the *Sequence Generator* when generating the next word (x_i) in the sequence (More details in section 3.5).

Here, we assume that the influence vector γ_T corresponding to current timestamp T, can be approximated from the historical topic distribution $\theta_{T-r:T-1}$. This assumption is reasonable, because most text stream data do not evolve dramatically

over night, rather their topical shift happens quite gradually. Take for example some particular research community like SIGKDD. The topic distribution in papers published in a particular year is unlikely dramatically different from the previous two years, rather they are somewhat correlated.

Mathematically, let θ_T denote the topic distribution for the generated text at timestamp T, then the function of *Influence Generator* is expressed as follows ($\|$ means the concatenation operation):

$$\theta_{T-r:T-1} = \theta_{T-r} \| \theta_{T-r+1} \| .... \| \theta_{T-1} \quad (4)$$

$$\begin{aligned} \gamma_T &= \Gamma(\theta_{T-r:T-1}) \\ &= W_2^\Gamma \cdot \left[ReLU(W_1^\Gamma \cdot \theta_{T-r:T-1} + B_1^\Gamma) \right] + B_2^\Gamma \end{aligned} \quad (5)$$

3.5 TILM as a Unified Model

Now that we have presented the three building blocks of TILM, this section presents how these different components interact with each other and work as a unified architecture to model stream text data by capturing the fluctuations of topical influence over time. The process can be thought of as generating text corresponding to a particular timestamp T. Thus, the whole process starts with a timestamp T as input and the start-of-sentence marker (let's call it #) as the sequence generated so far. The next task is to generate one word at a time iteratively until the end-of-sentence marker is generated (let's call it *). The exact process of generating the next word y_i in the sequence is demonstrated in Figure 1.

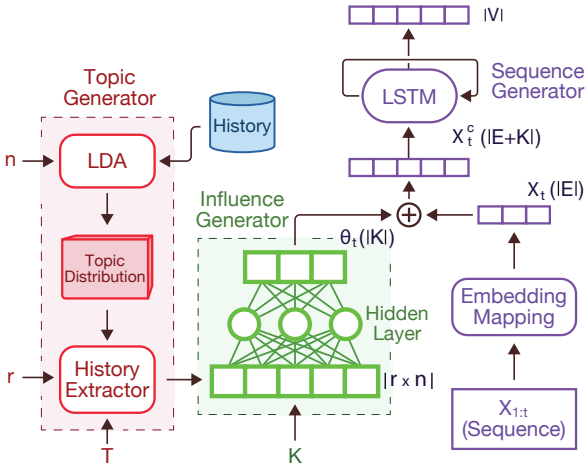


Figure 1: TILM architecture: Topic Generator (Red), Influence Generator (Green) and Sequence Generator (Purple)

The first step of generation process is to infer n different topics from a related (possibly same) historical text stream and compute the topic distributions for each unique timestamp observed in the training data. Next, given a particular timestamp T as input, the *History Extractor* Module extracts the historical topic distributions corresponding to previous r timestamps and concatenates them to generate a vector representation of the history, i.e., $\theta_{T-r:T-1}$ of dimension $r \times n$ (see the previous subsection for details). $\theta_{T-r:T-1}$ is then passed through *Influence Generator* Γ which outputs the K dimensional influence vector γ_T. For a particular timestamp T, γ_T is fixed and can be re-used for any text generation task tied to the timestamp T. The bottom middle section (Green Color) of Figure 1 shows the feed-forward neural network of *Influence Generator*.

The next trick in TILM is to concatenate the influence vector (γ_T) with the vector representation of each word in the sequence generated so far. This means, for each word in $\{x_1, x_2, ..., x_i\}$, γ_T is concatenated to each of their vector representations to create an augmented representation $\{x_1^C, x_2^C, ..., x_i^C\}$, i.e., while generating the next word $x_{i+1} = y_i$ in the sequence, all the previous words in the sequence share the same topical influence represented by vector γ_T. This augmented representation essentially allows TILM to capture the dynamic nature of text stream data as the influence vector injects evolving topical influence into the generation process. Finally, the augmented representations $\{x_1^C, x_2^C, ..., x_i^C\}$ are fed into the recurrent neural network model to compute a hidden state h_i . The final output vector Y is computed by applying a linear transformation on h_i. Note that, vector Y is a real-valued vector. We apply a Softmax function on Y to convert it into a valid probability distribution, sampling from which, the next word in the sequence is generated. The mathematical formulas behind the entire generation process is summarized below:

$$Y = W^\Omega \cdot h_i + B^\Omega, \quad (6)$$

$$h_i = \Omega(h_{i-1}, x_i^C), \quad (7)$$

$$x_i^C = x_i \| \gamma_T. \quad (8)$$

Thus, Y can be written as follows:

$$Y = W^\Omega \cdot \Omega(h_{i-1}, x_i \| \gamma_T) + B^\Omega. \quad (9)$$

Here, γ_T is obtained as follows:

$$\gamma_T = W_2^\Gamma \cdot \left[ReLU(W_1^\Gamma \cdot \theta_{T-r:T-1} + B_1^\Gamma) \right] + B_2^\Gamma. \quad (10)$$

Here, $\theta_{T-r:T-1}$ is the concatenation of topic distribution vector from previous r timestamps.

Finally, we apply a Softmax function on the output vector Y to convert it into a valid probability distribution $P(y_i|h_i)$, as follows:

$$P(y_i = v|h_i) = \frac{\exp(Y_i)}{\sum_{j=1}^{|V|} \exp(Y_j)}. \quad (11)$$

The next word $y_i = x_{i+1}$ in the sequence is generated by sampling from this conditional distribution. TILM repeats this whole process multiple times to generate new words in the sentence until a end-of-sentence marker is generated. The whole process is summarized in Algorithm 1, which describes the generation of a single sentence by TILM for a particular timestamp T.

Algorithm 1: Topical Influence Language Model (TILM)

Process TILM $(T, \Theta, \Gamma, \Omega, r, n, K, E)$;
Input : T: discrete timestamp
Θ: Topic Generator (n: number of topics)
Γ: Influence Generator (K: cardinality of influence vector)
Ω: Sequence Generator (E: cardinality of word vector)
r: History Window
Output: Generated sentence X corresponding to T
$x \leftarrow$ {start of sentence marker}
$\theta_{T-r:T-1} \leftarrow \Theta(\theta_{T-r}) \| \Theta(\theta_{T-r+1}) \| ... \| \Theta(\theta_{T-1})$, Topic History
$\gamma_T \leftarrow \Gamma(\theta_{T-r:T-1})$, Generate Influence Vector for timestamp T
$i \leftarrow 1$
repeat
  for $j \leftarrow 1$ **to** i **do**
    $x_j^C \leftarrow x_j \| \gamma_T$, where, $\|$ is concatenation operation
  end
  compute $h_i \leftarrow \Omega(x_{1:i}^C)$ by applying Ω recursively
  Draw word $y_i \sim P(y_i|h_i)$, where $P(y_i|h_i) \propto \exp(W^\Omega h_i + B^\Omega)$
  $x \leftarrow x \cup \{y_i\}$
  $i \leftarrow i + 1$
until *end of sentence marker is generated*;
return x

3.6 Estimation of TILM parameters

In this section, we present the estimation techniques for the optimal values of TILM model parameters. Close observation of Equation 6-10 reveals that TILM contains the following set of parameters:

$$\boldsymbol{W} = \left\{W^\Omega, B^\Omega, W_1^\Gamma, B_1^\Gamma, W_2^\Gamma, B_2^\Gamma\right\} \quad (12)$$

We find the optimal values for the parameter set $\boldsymbol{W}$ by maximizing the log-likelihood of the training text stream data. The optimization problem thus can be written as follows:

$$\boldsymbol{W}^* = \underset{W}{argmax} \quad \log L(x_1 x_2 ... x_n|W) \quad (13)$$

As maximizing the log-likelihood is the same as minimizing the negative of the log-likelihood function and as we know the exact word which comes next in the sequence during the training process, our optimization problem boils down to minimizing the softmax cross entropy with logits between the conditional distribution $P(y_i|h_i)$ and the one-hot encoding of the actual word that appears next in the training data. Softmax Cross Entropy with logits essentially measures the probability error in discrete classification tasks in which the classes are mutually exclusive. Thus,

$$\boldsymbol{W}^* = \underset{W}{argmin} \{-\log L(x_1 x_2 ... x_n|W)\} \quad (14)$$

$$= \underset{W}{argmin} \left\{ -\sum_{i=1}^{N} \sum_{v \in V} I(x_i, v) \cdot \log P(x_i = v|h_i(W)) \right\} \quad (15)$$

Here, N is the total number of words in the training data. $I(p, q)$ is an indicator function that returns 1 if $p = q$ and 0 otherwise.

We used back-propagation to learn the weights of the network connection edges of TILM. Specifically, we used Adaptive Moment Estimation, which is a popular stochastic gradient descent technique and commonly known as Adam Optimizer, to compute the gradient for minimizing our objective function in Equation 15. Adam Optimizer is an update to the RMSProp (Hinton et al., 2012), which is another popular optimizer. For more details, refer to (Kingma and Ba, 2014).

4 Experimental Design

4.1 Dataset

We experimented with six different sets of publication title stream data to evaluate the performance of TILM. These datasets were collected from the Open Academic Graph[1](Tang et al., 2008; Sinha et al., 2015). Here, we focused on studying how the paper titles published by different machine learning related conferences evolved over time. As community, we considered both the core machine learning community, e.g., NIPS, ICML as well as research communities that apply a fair share of machine learning, e.g, KDD, SIGIR. Specifically, we considered all the titles of papers published during the years 1996-2015 by the following six conference venues: NIPS, CVPR, ICML, KDD, SIGIR and WWW. For these datasets, the discrete timestamp corresponds to a particular year.

[1] https://www.openacademic.ai/oag/

Conf.	# of Titles	Title/ Year	Total Words	Words/ Title
KDD	5,499	274.95	49,980	9.08
NIPS	6,229	311.45	49,792	7.99
SIGIR	3,994	199.7	34,892	8.73
ICML	4,106	205.3	34,159	8.32
WWW	5,701	285.05	50,253	8.82
CVPR	10,121	506.05	90,890	8.98

Table 1: Dataset Summary

Each row in these datasets consists of a tuple <timestamp, paper-title> and altogether they contain $35,650$ paper titles in total. Again, each paper title can contribute multiple instances for predicting the next word which resulted in $309,966$ total instances (see Table 1).

4.2 Evaluation Roadmap

For a proper evaluation of TILM, we need a goal task which is both time-sensitive and requires influence modeling. To address this challenge, we evaluate TILM using a text forecasting task, where we attempt to predict the text content at future timestamps in a text stream based on the past data from related (including the same) streams.

As the setup of such a text forecasting task is essentially similar to summarizing *future* text data, we use two popular evaluation metrics from the literature of text summarization, i.e., BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), where a score is generated by comparing the automatically generated text against some reference text written by humans. However, neither BLEU nor ROUGE considers the notion of time, thus we need a time-sensitive customization of both BLEU and ROUGE. The simplest way to do this is to compare a TILM generated text for timestamp T against the original text corresponding to the same timestamp T. This means, when TILM is asked to generate a paper title relevant to timestamp T, the ground truth paper title against which the generated title is compared must also correspond to the timestamp T. Another challenge is to match the generated title against multiple independent publication titles corresponding to timestamp T. We address it by adopting a simple greedy approach where the TILM-generated title is matched against each ground truth title corresponding to the timestamp T and paired with the most similar one in terms of BLEU or ROUGE score. The matched ground truth title is then removed from the corpus so that the next generated title cannot match with the previously matched title again. This ensures that TILM is generating a diverse set of titles rather than just memorizing one single title from each timestamp T. This way, we can use TILM to generate multiple sentences (titles) for a particular timestamp T and then average the scores of all generated sentences to get an evaluation score corresponding to timestamp T. This whole computation process is presented in Algorithm 2, where we demonstrate the case for time-sensitive BLEU score. The case for ROUGE is exactly similar. Finally, these average scores across different timestamps can be further averaged to compute the overall forecasting score[2].

Algorithm 2: Time aware BLEU score computation

Time aware BLEU (T, G, R);
Input : T: discrete timestamp
G: Generated Text set
R: Reference Text set
Output: Time sensitive BLEU
$score \leftarrow 0$
$|G| \leftarrow$ number of sentences in G
for *each sentence* g *in* G **do**
  for *each sentence* r *in* R **do**
    compute $BLEU(g, r)$
  end
  $r^* = \underset{r}{argmax} \;\; BLEU(g, r)$
  $score \leftarrow score + BLEU(g, r^*)$
  $R \leftarrow R - \{r^*\}$
end
return $\frac{score}{|G|}$

4.3 Baseline Methods

The main questions we want to answer in our experiments are whether the incorporation of topical influence (from a related stream) is beneficial for language modeling and whether the specific configuration of TILM we described earlier is an effective way to capture evolving topical influence. To answer the second question, we compare TILM with two baselines which are both variants of TILM but with different ways to capture influence. The first one is called *RILSTM* which is identical to TILM except that the influence vector of RILSTM is generated randomly as opposed to generating it by the *Influence Generator* of TILM. The second baseline is called *IILSTM* where we do not inject the influence vector as a bias into

[2]All the codes and evaluation scripts for experimentation can be found at the following link: (https://bitbucket.org/karmake2/tilm/src/master/

Acronym	Details	Nature
Bigram	Bigram Language Model	Static
LSTM	Long short-term memory	Static
RILSTM	LSTM with Random Influence	Dynamic
IILSTM	Sampling from Joint LSTM-Influence Distribution	Dynamic
TILM	Topical Influence LM	Dynamic

Table 2: Baselines for Quantitative Comparison.

the vector representation of words, rather, the *Influence Generator* directly computes a probability distribution for sampling the next word and this probability is multiplied with the probability computed independently by LSTM. To answer the first question, i.e., to verify the benefit of modeling the evolution of topical influence, we also compared TILM against two baselines representing static models: simple bigram language model and Long-Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997; Gers et al., 1999). Table 2 contains the summary of these baseline algorithms along with TILM.

5 Results

This section presents both quantitative and qualitative evaluation results for TILM. For all the results reported in this section, TILM used the following parameter settings: r was set to 3, for both *Sequence Generator* and *Influence Generator*, the number of hidden units was empirically set to 256, K (dimension of influence vector) was set to 15, batch size was set to 2000 instances and the learning rate was set to 0.01. For the *Topic Generator*, n (number of topics) was set to 15, θ_T was computed from target text stream itself and LDA was run using $\alpha = 0.1$ and $\beta = 0.05$.

5.1 Quantitative Evaluation

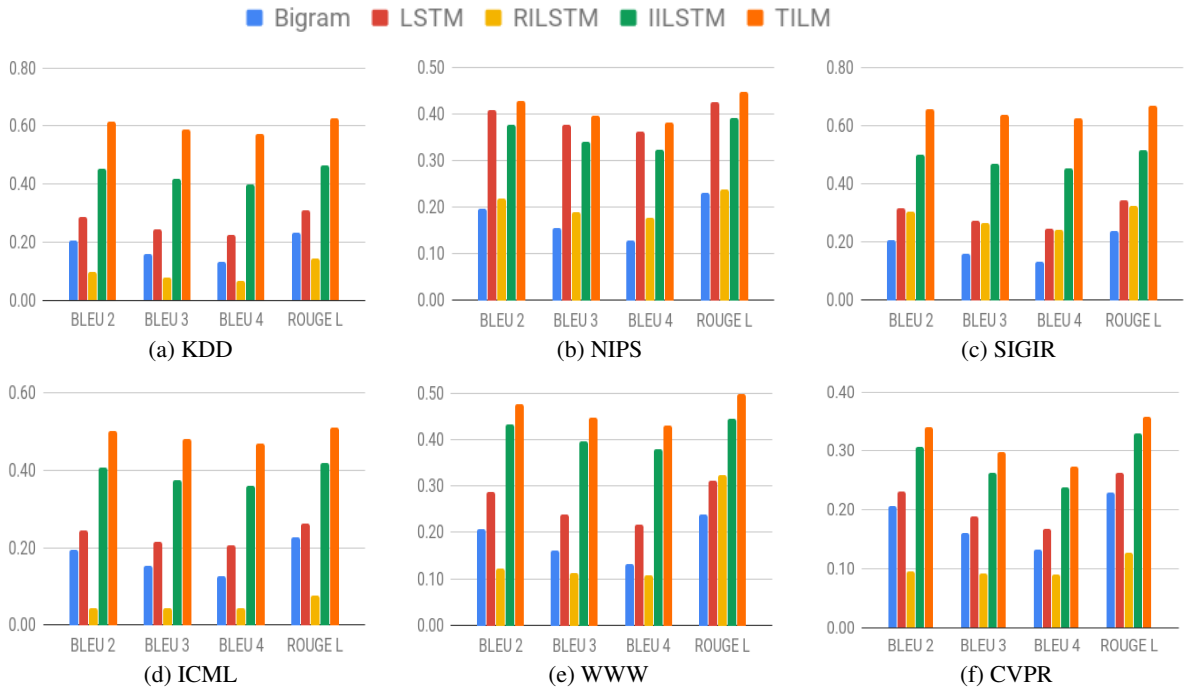
Figure 2 provides the summary of results for comparing TILM against multiple baseline methods. Close examination of Figure 2 reveals that TILM outperforms all other baselines by a clear margin for all six datasets. For example, BLEU-4 score obtained by TILM on KDD Dataset is 0.57, while LSTM obtained only a score of 0.22. ROUGE-L score obtained by TILM is 0.63, while it is 0.31 for LSTM. This clearly indicates that TILM can indeed capture the temporal evolution of KDD paper titles over time and given a input timestamp T, can generate text relevant to T. Also note that, RILSTM performs significantly worse compared to LSTM for most datasets which implies that the influence vector plays the key role in helping TILM capture the evolution of the text stream. It is also noteworthy that IILSTM is the second best performing method which confirms that injecting influence vector as a bias into the word representation works better than using the Joint LSTM-Influence distribution obtained by simply multiplying influence probabilities with LSTM probabilities.

To get more insights into the performance of TILM, we plot the timestamp-wise performance of all compared methods for KDD Dataset (BLEU 4) and WWW Dataset (ROUGE L) in Figure 3 [other plots are similar and omitted due to lack of space]. A general inspection of Figure 3 also demonstrates the superiority of TILM for the text forecasting task, where, for any performance metric, TILM obtains the best score across different timestamps for most of the cases.

External Influence: So far, we have only considered the influence of the community for which the text generation is targeted towards. However, other related communities also pose indirect influence on the text content generated within the target community. For example, a shift in the interest of theoretical machine learning conferences like ICML often influences the research directions pursued by more applied conferences like KDD or WWW. To test this hypothesis, we conducted a series of experiments where instead of using the influence of the target community (e.g., KDD, WWW) itself, we computed the influence vector from the historical topic distribution of a core machine learning community (e.g. ICML, NIPS). We call this approach TILM-EI where EI means external influence. We conducted another set of experiments where we computed influence vectors from both the target community and a related external community and injected both influence vectors into the TILM process. We call this approach TILM-CI where CI stands for combined influence. Two sample results from these experiments are shown in Figure 4, i.e., influence of ICML on KDD and WWW, respectively. Experiments results suggest that, although TILM-EI is not always better than TILM itself, TILM-CI outperforms basic TILM as TILM-CI combines both internal and external influence.

5.2 Qualitative Evaluation

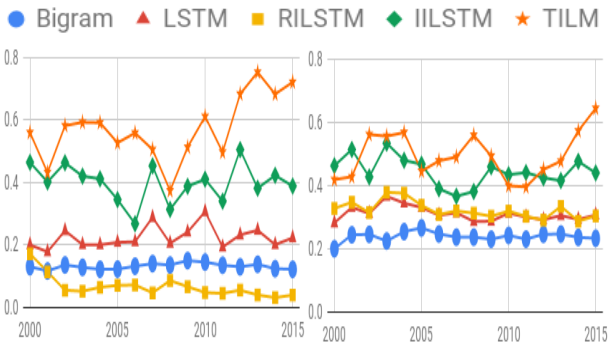
In this section, we present qualitative results to show the great potentials of TILM. We first ran LDA on paper titles from SIGIR and KDD over the year range 2000-2015. Number of topics was



(a) KDD (b) NIPS (c) SIGIR

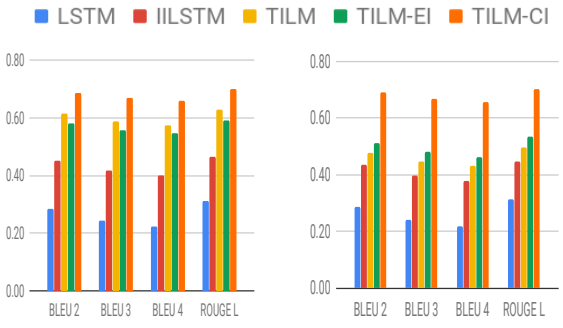
(d) ICML (e) WWW (f) CVPR

Figure 2: Comparison of TILM against baselines for text forecasting.



(a) KDD: BLEU 4 (b) WWW: ROUGE L

Figure 3: Year-Wise Performance distribution of TILM against different baseline text generation techniques



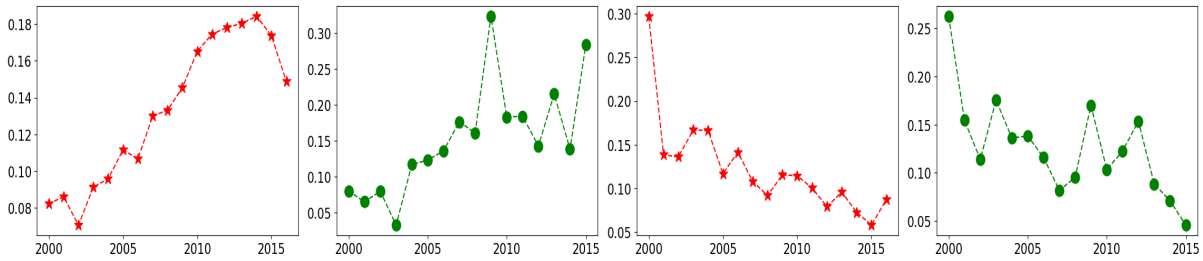
(a) ICML influencing KDD (b) ICML influencing WWW

Figure 4: Results of adding external influence into TILM for text forecasting task.

Topic	Top Keywords
Optimization	matrix, gradient, sparse, convex, stochastic
Search Relevance	information, retrieval, search, index, document
Rule Mining	rule, discovery, association, pattern, mine
Social Networks	social, network, recommender, community, topic
SVM Classifiers	supervised, learning, support, vector, machine
User Behavior Modeling	log, behavior, personalization, click, feedback

Table 3: Sample topics Extracted from KDD and SIGIR for year range [1995-2015] Using LDA

set to 15. Table 3 shows six example topics along with top 5 keywords for each topic. Next, we did some topic trend analysis for SIGIR conference in Figure 5. For SIGIR, we considered two topics, i.e., *User Behavior Modeling* [Figure 5a, 5b] and *Search and Relevance* [Figure 5c, 5d]. Beside plotting the original topic-distribution trend computed from the real conference proceedings (Figures in red color), we also plot the simulated topic-distribution trend computed from the text generated by TILM (Figures in green color). Close observation of Figure 5 confirms that TILM can indeed generate sentences aligned with the evolution of the text stream corresponding to evolving topical influence. For example, Figure 5a shows that research interest towards *User Behavior Modeling* grew significantly within SIGIR community in the past ten years, which is also nicely reflected in the text generated by TILM [Figure 5b]. On the other hand, research on Search and relevance almost matured after 2008 within the SIGIR community [Figure 5c], which has also been captured



(a) $SIGIR$(**Original**): User Behavior Modeling (b) $SIGIR$(**Generated**): User Behavior Modeling (c) $SIGIR$(**Original**): Search & Relevance (d) $SIGIR$(**Generated**): Search & Relevance

Figure 5: Topic trend analysis to demonstrate how TILM captures evolution of SIGIR research paper titles.

year	#	Sample Generated Title
2000 - 2002	1	discovering in hierarchical rules using lexical knowledge
	2	data mining criteria for tree based regression and classification
	3	mining frequent class sets in spatial databases
2007 - 2009	1	a framework for community identification in dynamic social networks
	2	learning preferences of new users in recommender systems
	3	data mining for intrusion detection from outliers
2012 - 2015	1	deep model based transfer and multi task learning for biological image analysis
	2	a bayesian framework for estimating properties of information network
	3	active learning for sparse bayesian classification

Table 4: Sample titles generated by TILM for conference KDD across different year ranges

effectively by TILM and apparent from the decaying trend of Figure 5d.

Finally, Table 4 presents some sample paper titles generated by TILM for different time ranges targeted towards KDD community. A closer look into Table 4 reveals that TILM can generate syntactically correct, semantically coherent and time-sensitive evolving text. It is worth mentioning that, TILM did not store any paper-to-year mapping information. Table 4 also nicely captures the interest shift within KDD community over the years. For example, paper titles generated for year range 2000-2002 include topics like rule mining and tree based classifications, while paper titles generated for year range 2012-2015 include topics like deep learning and active learning, which is consistent with our expectation.

6 Conclusion

We studied how to improve neural language models by incorporating topical influence from contextual factors and proposed a novel Topical Influence Language Model (TILM), which includes a novel extension of a basic neural language model by incorporating both a topic generator (based on topic modeling) and a neural network-based influence modeling component, leading to a general architecture with three major components: *Sequence Generator*, *Topic Generator* and *Influence Generator*. We quantitatively evaluated TILM using a text forecasting task on six publication stream datasets and demonstrated that it is beneficial to incorporate topical influence in language modeling and TILM outperforms multiple baseline methods by a significant margin. As a novel language model, TILM allows for leveraging related text streams to improve accuracy of language modeling of a target stream, thus potentially helping improve many applications where language models are applied. We also show that TILM is able to generate well-structured, meaningful text content corresponding to future time stamps, thus potentially allowing us to predict topical trends in the future, which would be useful for optimizing decision making. Moreover, we also showed the potential that TILM can be used as a tool to compare influences of different external streams on a particular target stream, thus facilitating cross-stream influence analysis. While it is not the focus of this paper, such analysis clearly opens up an interesting direction for analyzing multiple text streams to understand their influences in a topic-specific way, a highly promising direction for future research.

7 Acknowledgement

We sincerely thank Hao Ma, Bin Bi, Kuansan Wang and Jiawei Han for their valuable feedback during the initial phase of generating the idea for this paper. We also thank the three anonymous reviews for their unanimous recognition of the importance of this new research direction which would help promoting this direction to a greater extent.

References

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Tong Che, Yanran Li, Ruixiang Zhang, R. Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *CoRR*, abs/1702.07983.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Adji B Dieng, Chong Wang, Jianfeng Gao, and John Paisley. 2016. Topicrnn: A recurrent neural network with long-range semantic dependency. *arXiv preprint arXiv:1611.01702*.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.

G Hinton, N Srivastava, and K Swersky. 2012. Rmsprop: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Shubhra Kanti Karmaker Santu, Liangda Li, Yi Chang, and ChengXiang Zhai. 2018. Jim: Joint influence modeling for collective search behavior. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 637–646. ACM.

Shubhra Kanti Karmaker Santu, Liangda Li, Dae Hoon Park, Yi Chang, and Chengxiang Zhai. 2017. Modeling the influence of popular trending events on user search behavior. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 535–544. International World Wide Web Conferences Steering Committee.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. 2014. Multimodal neural language models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 595–603.

Karen Kukich. 1987. *Where do Phrases Come from: Some Preliminary Experiments in Connectionist Phrase Generation*, pages 405–421. Springer Netherlands, Dordrecht.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. *CoRR*, abs/1705.11001.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2014. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 5528–5531.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Joseph Pal, and Aaron C. Courville. 2017. Adversarial generation of natural language. *CoRR*, abs/1705.10929.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th*

international conference on world wide web, pages 243–246. ACM.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1017–1024.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. *arXiv preprint arXiv:1706.03850*.

Pretraining-Based Natural Language Generation for Text Summarization

Haoyu Zhang[†], **Jingjing Cai**[‡], **Jianjun Xu**[†], **Ji Wang**[†]
[†]HPCL, College of Computer, National University of Defense Technology, China
[‡]Dalian Neusoft University of Information, China
{zhanghaoyu10, jjxu, wj}@nudt.edu.cn
caijingjing@neusoft.edu.cn

Abstract

In this paper, we propose a novel pretraining-based encoder-decoder framework, which can generate the output sequence based on the input sequence in a two-stage manner. For the encoder of our model, we encode the input sequence into context representations using BERT. For the decoder, there are two stages in our model, in the first stage, we use a Transformer-based decoder to generate a draft output sequence. In the second stage, we mask each word of the draft sequence and feed it to BERT, then by combining the input sequence and the draft representation generated by BERT, we use a Transformer-based decoder to predict the refined word for each masked position. To the best of our knowledge, our approach is the first method which applies the BERT into text generation tasks. As the first step in this direction, we evaluate our proposed method on the text summarization task. Experimental results show that our model achieves new state-of-the-art on both CNN/Daily Mail and New York Times datasets.

1 Introduction

Text summarization generates summaries from input documents while keeping salient information. It is an important task and can be applied to several real-world applications. Many methods have been proposed to solve the text summarization problem (See et al., 2017; Nallapati et al., 2017; Zhou et al., 2018; Gehrmann et al., 2018). There are two main text summarization techniques: extractive and abstractive. Extractive summarization generates summary by selecting salient sentences or phrases from the source text, while abstractive methods paraphrase and restructure sentences to compose the summary. We focus on abstractive summarization in this work as it is more flexible and thus can generate more diverse summaries.

Recently, many abstractive approaches are introduced based on neural sequence-to-sequence framework (Paulus et al., 2018; See et al., 2017; Gehrmann et al., 2018; Li et al., 2018). Based on the sequence-to-sequence model with copy mechanism, (See et al., 2017) incorporates a coverage vector to track and control attention scores on source text. (Paulus et al., 2018) introduce intra-temporal attention processes in the encoder and decoder to address the repetition and incoherent problem.

There are two issues in previous abstractive methods: 1) these methods use left-context-only decoder, thus do not have complete context when predicting each word. 2) they do not utilize the pre-trained contextualized language models on the decoder side, so it is more difficult for the decoder to learn summary representations, context interactions and language modeling together.

Recently, BERT has been successfully used in various natural language processing tasks, such as textual entailment, name entity recognition and machine reading comprehensions. In this paper, we present a novel natural language generation model based on pre-trained language models (we use BERT in this work). As far as we know, this is the first work to extend BERT to the sequence generation task. To address the above issues of previous abstractive methods, in our model, we design a two-stage decoding process to make good use of BERT's context modeling ability. On the first stage, we generate the summary using a left-context-only-decoder. On the second stage, we mask each word of the summary and predict the refined word one-by-one using a refine decoder. To further improve the naturalness of the generated sequence, we cooperate reinforcement objective with the refine decoder.

The main contributions of this work are:

1. We propose a natural language generation

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 789–797
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

model based on BERT, making good use of the pre-trained language model in the encoder and decoder process, and the model can be trained end-to-end without handcrafted features.

2. We design a two-stage decoder process. In this architecture, our model can generate each word of the summary considering both sides' context information.

3. We conduct experiments on the benchmark datasets CNN/Daily Mail and New York Times. Our model achieves a 33.48 average of ROUGE-1, ROUGE-2 and ROUGE-L on the CNN/Daily Mail, which is state-of-the-art. On the New York Times dataset, our model achieves about 5.6% relative improvement over ROUGE-1.

2 Background

2.1 Text Summarization

In this paper, we focus on single-document multi-sentence summarization and propose a supervised abstractive model based on the neural attentive sequence-to-sequence framework which consists of two parts: a neural network for the encoder and another network for the decoder. The encoder encodes the input sequence to intermediate representation and the decoder predicts one word at a time step given the input sequence representation vector and previous decoded output. The goal of the model is to maximize the probability of generating the correct target sequences. In the encoding and generation process, the attention mechanism is used to concentrate on the most important positions of text. The learning objective of most sequence-to-sequence models is to minimize the negative log likelihood of the generated sequence as shown in following equation, where y_t^* is the t-th ground-truth summary token.

$$Loss = -\log \sum_{t=1}^{|y|} P(y_t^* | y_{<t}^*, X) \tag{1}$$

However, with this objective, traditional sequence generation models consider only one direction context in the decoding process, which could cause performance degradation since complete context of one token contains preceding and following tokens, thus feeding only preceded decoded words to the decoder so that the model may generate unnatural sequences. For example, attentive sequence-to-sequence models often generate sequences with repeated phrases which harm the naturalness. Some previous works mitigate this problem by improving the attention calculation process, but in this paper we show that feeding bi-directional context instead of left-only-context can better alleviate this problem.

2.2 Bi-Directional Pre-Trained Context Encoders

Recently, context encoders such as ELMo, GPT, and BERT have been widely used in many NLP tasks. These models are pre-trained on a huge unlabeled corpus and can generate better contextualized token embeddings, thus the approaches built on top of them can achieve better performance.

Since our method is based on BERT, we illustrate the process briefly here. BERT consists of several layers. In each layer there is first a multi-head self-attention sub-layer and then a linear affine sub-layer with the residual connection. In each self-attention sub-layer the attention scores e_{ij} are first calculated as Eq. (2) (3), in which d_e is output dimension, and W^Q, W^K are parameter matrices.

$$a_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d_e}} \tag{2}$$

$$e_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{N} \exp e_{ik}} \tag{3}$$

Then the output is calculated as Eq. (4), which is the weighted sum of previous outputs h added by previous output h_i. The last layer outputs is context encoding of input sequence.

$$o_i = h_i + \sum_{j=1}^{N} e_{ij}(h_j W_V) \tag{4}$$

Despite the wide usage and huge success, there is also a mismatch problem between these pre-trained context encoders and sequence-to-sequence models. The issue is that while using a pre-trained context encoder like BERT, they model token-level representations by conditioning on both direction context. During pre-training, they are fed with complete sequences. However, with a left-context-only decoder, these pre-trained language models will suffer from incomplete and inconsistent context and thus cannot generate good enough context-aware word representations, especially during the inference process.

3 Model

In this section, we describe the structure of our model, which learns to generate an abstractive multi-sentence summary from a given source document.

Based on the sequence-to-sequence framework built on top of BERT, we first design a refine decoder at word-level to tackle the two problems described in the above section. We also introduce a discrete objective for the refine decoders to reduce the exposure bias problem. The overall structure of our model is illustrated in Figure 1.

3.1 Problem Formulation

We denote the input document as $X = \{x_1, \ldots, x_m\}$ where $x_i \in \mathcal{X}$ represents one source token. The corresponding summary is denoted as $Y = \{y_1, \ldots, y_L\}$, L represents the summary length.

Given input document X, we first predict the summary draft by a left-context-only decoder, and then using the generated summary draft we can condition on both context sides and refine the content of the summary. The draft will guide and constrain the refine process of summary.

3.2 Summary Draft Generation

The summary draft is based on the sequence-to-sequence model. On the encoder side the input document X is encoded into representation vectors $H = \{h_1, \ldots, h_m\}$, and then fed to the decoder to generate the summary draft $A = \{a_1, \ldots, a_{|a|}\}$.

3.2.1 Encoder

We simply use BERT as the encoder. It first maps the input sequence to word embeddings and then computes document embeddings as the encoder's output, denoted by following equation.

$$H = BERT(x_1, \ldots, x_m) \tag{5}$$

3.2.2 Summary Draft Decoder

In the draft decoder, we first introduce BERT's word embedding matrix to map the previous summary draft outputs $\{y_1, \ldots, y_{t-1}\}$ into embeddings vectors $\{q_1, \ldots, q_{t-1}\}$ at t-th time step. Note that as the input sequence of the decoder is not complete, we do not use the BERT network to predict the context vectors here.

Then we introduce an N layer Transformer decoder to learn the conditional probability $P(A|H)$. Transformer's encoder-decoder multi-head attention helps the decoder learn soft alignments between summary and source document. At the t-th time step, the draft decoder predicts output probability conditioned on previous outputs and encoder hidden representations as shown in Eq. (6), in which $q_{<t} = \{q_1, \ldots, q_{t-1}\}$. Each generated sequence will be truncated in the first position of a special token '[PAD]'. The total summary draft decoder progress is shown in Stage 1 of Figure 1.

$$P_t^{vocab}(w) = f_{dec}(q_{<t}, H) \tag{6}$$

$$L_{dec} = \sum_{t=1}^{|a|} -\log P(a_t = y_t^* | a_{<t}, H) \tag{7}$$

As Eq. (7) shows, the decoder's learning objective is to minimize negative likelihood of conditional probability, in which y_t^* is the t-th ground truth word of summary.

However a decoder with this structure is not sufficient enough: if we use the BERT network in this decoder, then during training and inference, in-complete context(part of sentence) is fed into the BERT module, and although we can fine-tune BERT's parameters, the input distribution is quite different from the pre-train process, and thus harms the quality of generated context representations.

If we just use the embedding matrix here, it will be more difficult for the decoder with fresh parameters to learn to model representations as well as vocabulary probabilities, from a relative small corpus compared to BERT's huge pre-training corpus. In a word, the decoder cannot utilize BERT's ability to generate high quality context vectors, which will also harm performance.

This issue exists when using any other contextualized word representations, so we design a refine process to mitigate it in our approach which will be described in the next sub-section.

3.2.3 Copy Mechanism

As some summary tokens are out-of-vocabulary words and occurs in input document, we incorporate copy mechanism (Gu et al., 2016) based on the Transformer decoder, we will describe it briefly.

At decoder time step t, we first calculate the attention probability distribution over source document X using the bi-linear dot product of the last

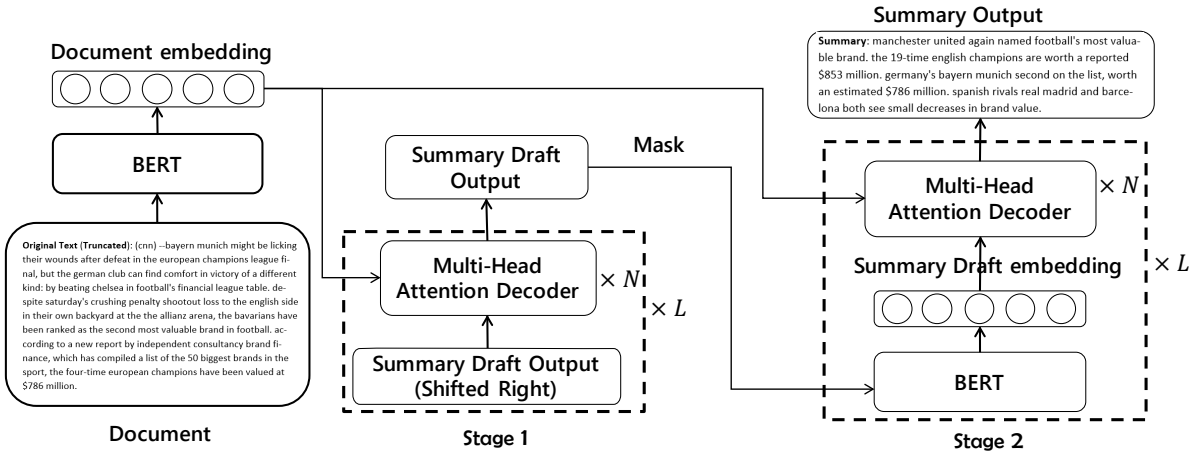


Figure 1: Model Overview, N represents decoder layer number and L represents summary length.

layer decoder output of Transformer o_t and the encoder output h_j, shown in Eq. (8) (9).

$$u_t^j = o_t W_c h_j \tag{8}$$

$$\alpha_t^j = \frac{\exp u_t^j}{\sum_{k=1}^{N} \exp u_t^k} \tag{9}$$

We then calculate copying gate $g_t \in [0, 1]$, which makes a soft choice between selecting from source and generating from vocabulary, W_c, W_g, b_g are parameters:

$$g_t = sigmoid(W_g \cdot [o_t, h] + b_g) \tag{10}$$

Using g_t we calculate the weighted sum of copy probability and generation probability to get the final predicted probability of extended vocabulary $\mathcal{V} + \mathcal{X}$, where $\mathcal{X}$ is the set of out of vocabulary words from the source document. The final probability is calculated as follow:

$$P_t(w) = (1 - g_t) P_t^{vocab}(w) + g_t \sum_{i:w_i=w} \alpha_t^i \tag{11}$$

3.3 Summary Refine Process

The main reason to introduce the refine process is to enhance the decoder using BERT's contextualized representations, so we do not modify the encoder and reuse it during this process.

On the decoder side, we propose a new word-level refine decoder. The refine decoder receives a generated summary draft as input, and outputs a refined summary. As Figure 1 Stage 2 shows, it first masks each word in the summary draft one by one, then feeds the draft to BERT to generate context vectors. Finally it predicts a refined summary word using an N layer Transformer decoder which is the same as the draft decoder. At t-th time step the t-th word of input summary is masked, and the decoder predicts the refined word given other words of the summary.

The learning objective of this process is shown in Eq. (12), y_t is the t-th summary word and y_t^* for the ground-truth summary word, and $a_{\neq t} = \{a_1, \ldots, a_{t-1}, a_{t+1}, \ldots, a_{|y|}\}$.

$$L_{refine} = \sum_{t=1}^{|y|} -\log P(y_t = y_t^* | a_{\neq t}, H) \tag{12}$$

From the view of BERT or other contextualized embeddings, the refine decoding process provides a more complete input sequence which is consistent with their pre-training processes. Intuitively, this process works as follows: first the draft decoder writes a summary draft based on a document, and then the refine decoder edits the draft. It concentrates on one word at a time, based on the source document as well as other words.

We design the word-level refine decoder because this process is similar to the cloze task in BERT's pre-train process, therefore by using the ability of the contextual language model the decoder can generate more fluent and natural sequences.

The parameters are shared between the draft decoder and refine decoder, as we find that using individual parameters the model's performance degrades a lot. The reason may be that we use teach-forcing during training, and thus the word-level refine decoder learns to predict words given all the

other ground-truth words of summary. This objective is similar to the language model's pre-train objective, and is probably not enough for the decoder to learn to generate refined summaries. So in our model all decoders share the same parameters.

3.3.1 Mixed Objective

For summarization, ROUGE is usually used as the evaluation metric, however during model training the objective is to maximize the log likelihood of generated sequences. This mis-match harms the model's performance. Similar to previous work (Kryściński et al., 2018), we add a discrete objective to the model, and optimize it by introducing the policy gradient method. The discrete objective for the summary draft process is as shown in Eq. (13), where a^s is the draft summary sampled from predicted distribution, and $R(a^s)$ is the reward score compared with the ground-truth summary, we use ROUGE-L in our experiment. To balance between optimizing the discrete objective and generating readable sequences, we mix the discrete objective with maximum-likelihood objective. Eq. (14) shows the final objective for the draft process, note here L_{dec} is $-logP(a|x)$. In the refine process we introduce similar objectives.

$$L^{rl}_{dec} = R(a^s) \cdot [-\log(P(a^s|x))] \quad (13)$$

$$\hat{L}_{dec} = \gamma * L^{rl}_{dec} + (1-\gamma) * L_{dec} \quad (14)$$

3.4 Learning and Inference

During model training, the objective of our model is sum of the two processes, jointly trained using "teacher-forcing" algorithm. During training we feed the ground-truth summary to each decoder and minimize the following objective.

$$L_{model} = \hat{L}_{dec} + \hat{L}_{refine} \quad (15)$$

At test time, each time step we choose the predicted word by $\hat{y} = argmax_{y'} P(y'|x)$, use beam search to generate the draft summaries, and use greedy search to generate the refined summaries.

4 Experiment

4.1 Settings

In this work, all of our models are built on $BERT_{BASE}$, although another larger pre-trained model with better performance ($BERT_{LARGE}$) has published but it costs too much time and GPU memory. We use WordPiece embeddings with a 30,000 vocabulary which is the same as BERT. We set the layer of transformer decoders to 12(8 on NYT50), and set the attention heads number to 12(8 on NYT50), set fully-connected sub-layer hidden size to 3072. We train the model using an Adam optimizer with learning rate of $3e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-9}$ and use a dynamic learning rate during the training process. For regularization, we use dropout (Srivastava et al., 2014) and label smoothing (Szegedy et al., 2016) in our models and set the dropout rate to 0.15, and the label smoothing value to 0.1. We set the RL objective factor γ to 0.99.

During training, we set the batch size to 36, and train for 4 epochs(8 epochs for NYT50 since it has many fewer training samples), after training the best model are selected from last 10 models based on development set performance. Due to GPU memory limit, we use gradient accumulation, set accumulate step to 12 and feed 3 samples at each step. We use beam size 4 and length penalty of 1.0 to generate logical form sequences.

During inference, we filter repeated tri-grams in beam-search process by setting word probability to zero if it will generate an tri-gram which exists in the existing summary. It is a nice method to avoid phrase repetition since the two datasets seldom contains repeated tri-grams in one summary.

4.1.1 Datasets

To evaluate the performance of our model, we conduct experiments on CNN/Daily Mail dataset, which is a large collection of news articles and modified for summarization. Following (See et al., 2017) we choose the non-anonymized version of the dataset, which consists of more than 280,000 training samples and 11490 test set samples.

We also conduct experiments on the New York Times(NYT) dataset which also consists of many news articles. The original dataset can be applied here.[1] In our experiment, we follow the dataset splits and other pre-process settings of (Durrett et al., 2016). We first filter all samples without a full article text or abstract and then remove all samples with summaries shorter than 50 words. Then we choose the test set based on the date of publication(all examples published after January 1, 2007). The final dataset contains 22,000 train-

[1] http://duc.nist.gov/

Model	ROUGE-1	ROUGE-2	ROUGE-L	R-AVG
Extractive				
lead-3 (See et al., 2017)	40.34	17.70	36.57	31.54
SummmaRuNNer (Nallapati et al., 2017)	39.60	16.20	35.30	30.37
Refresh (Narayan et al., 2018)	40.00	18.20	36.60	31.60
DeepChannel (Shi et al., 2018)	41.50	17.77	37.62	32.30
rnn-ext + RL (Chen and Bansal, 2018)	41.47	18.72	37.76	32.65
MASK-LM^{global} (Chang et al., 2019)	41.60	19.10	37.60	32.77
NeuSUM (Zhou et al., 2018)	41.59	19.01	37.98	32.86
Abstractive				
PointerGenerator+Coverage (See et al., 2017)	39.53	17.28	36.38	31.06
ML+RL+intra-attn (Paulus et al., 2018)	39.87	15.82	36.90	30.87
inconsistency loss(Hsu et al., 2018)	40.68	17.97	37.13	31.93
Bottom-Up Summarization (Gehrmann et al., 2018)	41.22	18.68	38.34	32.75
DCA (Celikyilmaz et al., 2018)	41.69	19.47	37.92	33.11
Ours				
One-Stage	39.50	17.87	36.65	31.34
Two-Stage	41.38	19.34	38.37	33.03
Two-Stage + RL	**41.84**	**19.70**	**38.91**	**33.48**

Table 1: ROUGE F1 results for various models and ablations on the CNN/Daily Mail test set. R-AVG calculates average score of Rouge-1, Rouge-2 and Rouge-L.

ing samples and 3,452 test samples and is called NYT50 since all summaries are longer than 50 words.

We tokenize all sequences of the two datasets using the WordPiece tokenizer. After tokenizing, the average article length and summary length of CNN/Daily Mail are 691 and 51, and NYT50's average article length and summary length are 1152 and 75. We truncate the article length to 512, and the summary length to 100 in our experiment(max summary length is set to 150 on NYT50 as its average golden summary length is longer).

4.1.2 Evaluation Metrics

On CNN/Daily Mail dataset, we report the full-length F-1 score of the ROUGE-1, ROUGE-2 and ROUGE-L metrics, calculated using PyRouge package[2] and the Porter stemmer option. On NYT50, following (Paulus et al., 2018) we evaluate limited length ROUGE recall score(limit the generated summary length to the ground truth length). We split NYT50 summaries into sentences by semicolons to calculate the ROUGE scores.

4.2 Results and Analysis

Table 1 gives the results on CNN/Daily Mail dataset, we compare the performance of many recent approaches with our model. We classify them to two groups based on whether they are extractive or abstractive models. As the last line of the table lists, the ROUGE-1 and ROUGE-2 score of our full model is comparable with DCA, and outperforms on ROUGE-L. Also, compared to extractive models NeuSUM and MASK-LM^{global}, we achieve slight higher ROUGE-1. Except the above four scores, our model outperforms these models on all the other scores, and since we have 95% confidence interval of at most $\pm$ 0.20, these improvements are statistically significant. The test set outputs of our model are released for further study.[3]

4.2.1 Ablation Analysis

As the last four lines of Table 1 show, we conduct an ablation study on our model variants to analyze the importance of each component. We use three ablation models for the experiments. One-Stage: A sequence-to-sequence model with copy mechanism based on BERT; Two-Stage: Adding the refine decoder to the One-Stage model; Two-Stage

[2]pypi.python.org/pypi/pyrouge/0.1.3

[3]https://1drv.ms/u/s!AvPUdqbfQJ503Qffg8HZzV98iosq

+ RL: Full model with refine process cooperated with RL objective.

First, we compare the Two-Stage+RL model with Two-Stage ablation, we observe that the full model outperforms by 0.30 on average ROUGE, suggesting that the reinforcement objective helps the model effectively. Then we analyze the effect of refine process by removing it from the Two-Stage model, we observe that without the refine process the average ROUGE score drops by 1.69. The ablation study proves that each module is necessary for our full model, and the improvements are statistically significant on all metrics.

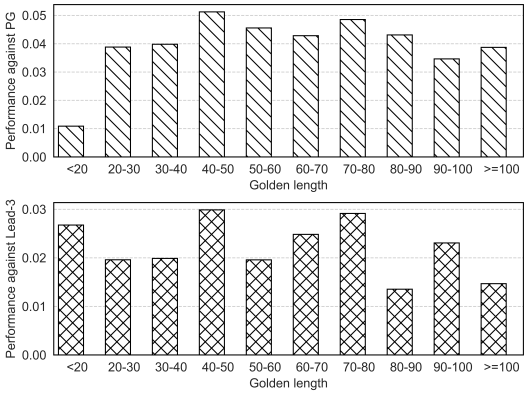


Figure 2: Average ROUGE-L improvement on CNN/Daily mail test set samples with different golden summary length.

4.2.2 Effects of Summary Length

To evaluate the impact of summary length on model performance, we compare the average rouge score improvements of our model with different length of ground-truth summaries. As the above sub-figure of Figure 2 shows, compared to Pointer-Generator with Coverage, on length interval 40-80(occupies about 70% of test set) the improvements of our model are higher than shorter samples, confirms that with better context representations, in longer documents our model can achieve higher performance.

As shown in the below sub-figure of Figure 2, compared to extractive baseline: Lead-3 (See et al., 2017), the advantage of our model will fall when golden summary length is greater than 80. This probably because that we truncate the long documents and golden summaries and cannot get full information, it could also because that the training data in these intervals is too few to train an abstractive model, so simple extractive method will not fall too far behind.

4.3 Additional Results on NYT50

Table 2 reports experiment results on the NYT50 corpus. Since the short summary samples are filtered, NYT50 has average longer summaries than CNN/Daily Mail. So the model needs to catch long-term dependency of the sequences to generate good summaries.

The first two lines of Table 2 show results of the two baselines introduced by (Durrett et al., 2016): these baselines select first n sentences, or select the first k words from the original document. Also we compare performance of our model with two recent models, we see 2.39 ROUGE-1 improvements compared to the ML+RL with intra-attn approach(previous SOTA) carries over to this dataset, which is a large margin. On ROUGE-2, our model also get an improvement of 0.51. The experiment proves that our approach can outperform competitive methods on different data distributions.

5 Related work

5.1 Text Summarization

Text summarization models are usually classified to abstractive and extractive ones. Recently, extractive models like DeepChannel (Shi et al., 2018), rnn-ext+RL (Chen and Bansal, 2018) and NeuSUM (Zhou et al., 2018) achieve higher performances using well-designed structures. For example, DeepChannel propose a salience estimation network and iteratively extract salient sentences. (Zhang et al., 2018) train a sentence compression model to teach another latent variable extractive model.

Also, several recent works focus on improving abstractive methods. (Gehrmann et al., 2018) design a content selector to over-determine phrases in a source document that should be part of the summary. (Hsu et al., 2018) introduce inconsistency loss to force words in less attended sentences(which determined by extractive model) to have lower generation probabilities. (Li et al., 2018) extend seq2seq model with an information selection network to generate more informative summaries.

Model	R-1	R-2
First sentences	28.60	17.30
First k words	35.70	21.60
Full (Durrett et al., 2016)	42.20	24.90
ML+RL+intra-attn (Paulus et al., 2018)	42.94	26.02
Two-Stage + RL (Ours)	**45.33**	**26.53**

Table 2: Limited length ROUGE recall results on the NYT50 test set.

5.2 Pre-trained language models

Pre-trained word vectors (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017) have been widely used in many NLP tasks. More recently, pre-trained language models (ELMo, GPT and BERT), have also achieved great success on several NLP problems such as textual entailment, semantic similarity, reading comprehension, and question answering (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018).

Some recent works also focus on leveraging pre-trained language models in summarization. (Radford et al., 2017) pretrain a language model and use it as the sentiment analyser when generating reviews of goods. (Kryściński et al., 2018) train a language model on golden summaries, and then use it on the decoder side to incorporate prior knowledge.

In this work, we use BERT(which is a pre-trained language model using large scale unlabeled data) on the encoder and decoder of a seq2seq model, and by designing a two stage decoding structure we build a competitive model for abstractive text summarization.

6 Conclusion and Future Work

In this work, we propose a two-stage model based on sequence-to-sequence paradigm. Our model utilize BERT on both encoder and decoder sides, and introduce reinforce objective in learning process. We evaluate our model on two benchmark datasets CNN/Daily Mail and New York Times, the experimental results show that compared to previous systems our approach effectively improves performance.

Although our experiments are conducted on summarization task, our model can be used in most natural language generation tasks, such as machine translation, question generation and paraphrasing. The refine decoder and mixed objective can also be applied on other sequence generation tasks, and we will investigate on them in future work.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep Communicating Agents for Abstractive Summarization. *arXiv preprint arXiv:1803.10357*.

Ming-Wei Chang, Kristina Toutanova, Kenton Lee, and Jacob Devlin. 2019. Language Model Pre-training for Hierarchical Document Representations. *arXiv preprint arXiv:1901.09128*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *ACL*.

Wan-Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. *arXiv preprint arXiv:1805.06266*.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. *arXiv preprint arXiv:1808.07913*.

Wei Li, Xinyan Xiao, Yajuan Lyu, and Yuanzhuo Wang. 2018. Improving Neural Abstractive Document Summarization with Explicit Information Selection Modeling. In *EMNLP*, pages 1787–1796.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3075–3081.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Ranking Sentences for Extractive Summarization with Reinforcement Learning. *arXiv preprint arXiv:1802.08636*.

Romain Paulus, Caiming Xiong, Richard Socher, and Palo Alto. 2018. A deep reinforced model for abstractive summarization. *ICLR*, pages 1–13.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 1073–1083.

Jiaxin Shi, Chen Liang, Lei Hou, Juanzi Li, Zhiyuan Liu, and Hanwang Zhang. 2018. DeepChannel: Salience Estimation by Contrastive Learning for Extractive Document Summarization. *CoRR*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pages 2818–2826.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. *arXiv preprint arXiv:1808.07187*.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 654–663.

Goal-Embedded Dual Hierarchical Model for Task-Oriented Dialogue Generation

Yi-An Lai
Amazon ML
Seattle
yianl@amazon.com

Arshit Gupta
Amazon ML
Seattle
arshig@amazon.com

Yi Zhang
Amazon ML
Seattle
yizhngn@amazon.com

Abstract

Hierarchical neural networks are often used to model inherent structures within dialogues. For goal-oriented dialogues, these models miss a mechanism adhering to the goals and neglect the distinct conversational patterns between two interlocutors. In this work, we propose *Goal-Embedded Dual Hierarchical Attentional Encoder-Decoder (G-DuHA)* able to center around goals and capture interlocutor-level disparity while modeling goal-oriented dialogues. Experiments on dialogue generation, response generation, and human evaluations demonstrate that the proposed model successfully generates higher-quality, more diverse and goal-centric dialogues. Moreover, we apply data augmentation via goal-oriented dialogue generation for task-oriented dialog systems with better performance achieved.

1 Introduction

Modeling a probability distribution over word sequences is a core topic in natural language processing, with language modeling being a flagship problem and mostly tackled via recurrent neural networks (RNNs) (Mikolov and Zweig, 2012; Melis et al., 2017; Merity et al., 2018).

Recently, dialogue modeling has drawn much attention with applications to response generation (Serban et al., 2016a; Li et al., 2016b; Asghar et al., 2018) or data augmentation (Yoo et al., 2019). It's inherently different from language modeling as the conversation is conducted in a turn-by-turn nature. (Serban et al., 2016b) imposes a hierarchical structure on encoder-decoder to model this utterance-level and dialogue-level structures, followed by (Serban et al., 2016c; Chen et al., 2018; Le et al., 2018a).

However, when modeling dialogues involving two interlocutors center around one or more goals, these systems generate utterances with the greatest likelihood but without a mechanism sticking to the goals. This makes them go off the rails and fail to model context-switching of goals. Most of the generated conversations become off-goal dialogues with utterances being non-relevant or contradicted to goals rather than on-goal dialogues. The differences are illustrated in Figure 1.

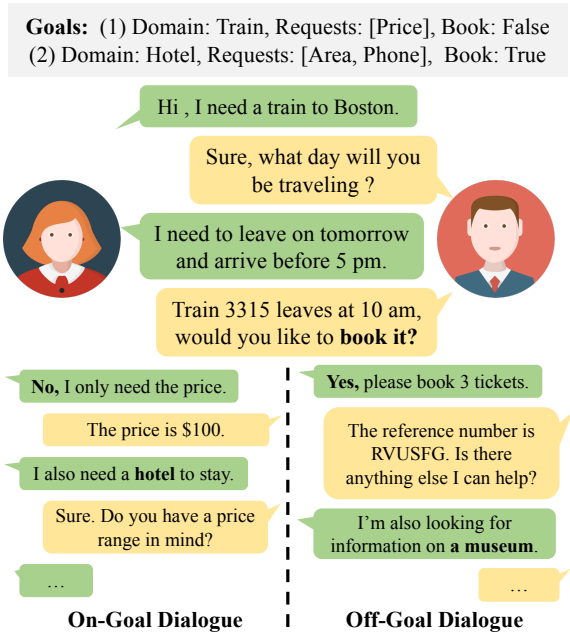


Figure 1: On-goal dialogues follow the given goals such as no booking of train tickets and a hotel reservation. Off-goal dialogues have context switches to other domains non-relevant to goals.

Besides, two interlocutors in a goal-oriented dialogue often play distinct roles as one has requests or goals to achieve and the other provides necessary support. Modeled by a single hierarchical RNN, this interlocutor-level disparity is neglected and constant context switching of roles could reduce the capacity for tracking conversational flow and long-term temporal structure.

To resolve the aforementioned issues when modeling goal-oriented dialogues, we propose

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 798–811
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

the Goal-Embedded Dual Hierarchical Attentional Encoder-Decoder (G-DuHA) to tackle the problems via three key features. First, the goal embedding module summarizes one or more goals of the current dialogue as goal contexts for the model to focus on across a conversation. Second, the dual hierarchical encoder-decoders can naturally capture interlocutor-level disparity and represent interactions of two interlocutors. Finally, attentions are introduced on word and dialogue levels to learn temporal dependencies more easily.

In this work, our contributions are that we propose a model called goal-embedded dual hierarchical attentional encoder-decoder (G-DuHA) to be the first model able to focus on goals and capture interlocutor-level disparity while modeling goal-oriented dialogues. With experiments on dialogue generation, response generation and human evaluations, we demonstrate that our model can generate higher-quality, more diverse and goal-focused dialogues. In addition, we leverage goal-oriented dialogue generation as data augmentation for task-oriented dialogue systems, with better performance achieved.

2 Related Work

Dialogues are sequences of utterances, which are sequences of words. For modeling or generating dialogues, hierarchical architectures are usually used to capture their conversational nature. Traditionally, language models are also used for modeling and generating word sequences. As goal-oriented dialogues are generated, they can be used in data augmentation for task-oriented dialogue systems. We review related works in these fields.

Dialogue Modeling. To model conversational context and turn-by-turn structure of dialogues, (Serban et al., 2016b) devised hierarchical recurrent encoder-decoder (HRED). Reinforcement and adversarial learning are then adopted to improve naturalness and diversity (Li et al., 2016b, 2017a). Integrating HRED with the latent variable models such as variational autoencoder (VAE) (Kingma and Welling, 2014) extends another line of advancements (Serban et al., 2016c; Zhao et al., 2017; Park et al., 2018; Le et al., 2018b). However, these systems are not designed for task-oriented dialogue modeling as goal information is not considered. Besides, conversations between two interlocutors are captured with a single encoder-decoder by these systems.

Language Modeling. A probability distribution of a word sequence $w_{1:T} = (w_1, w_2, ..., w_T)$ can be factorized as $p(w_1)\prod_{t=2}^{T} p(w_t|w_{1:t-1})$. To approximate the conditional probability $p(w_t|w_{1:t-1})$, counted statistics and smoothed N-gram models have been used before (Goodman, 2001; Katz, 1987; Kneser and Ney, 1995). Recently, RNN-based models have achieved a better performance (Mikolov et al., 2010; Józefowicz et al., 2016; Grave et al., 2017; Melis et al., 2018). As conversational nature is not explicitly modeled, models often have role-switching issues.

Task-Oriented Dialogue Systems. Conventional task-oriented dialog systems entails a sophisticated pipeline (Raux et al., 2005; Young et al., 2013) with components including spoken language understanding (Chen et al., 2016; Mesnil et al., 2015; Gupta et al., 2019), dialog state tracking (Henderson et al., 2014; Mrksic et al., 2017), and dialog policy learning (Su et al., 2016; Gašić and Young, 2014). Building a task-oriented dialogue agent via end-to-end approaches has been explored recently (Li et al., 2017b; Wen et al., 2017). Although several conversational datasets are published recently (Gopalakrishnan et al., 2019; Henderson et al., 2019), the scarcity of annotated conversational data remains a key problem when developing a dialog system. This motivates us to model task-oriented dialogues with goal information in order to achieve controlled dialogue generation for data augmentation.

3 Model Architecture

Given a set of goals and the seed user utterance, we want to generate a goal-centric or on-goal dialogue that follows the domain contexts and corresponding requests specified in goals. In this section, we start with the mathematical formulation, then introduce our proposed model, and describe our model's training objective and inference.

At training time, K dialogues $\{D_1, ..., D_K\}$ are given where each D_i associates with N_i goals $\mathbf{g_i} = \{g_{i1}, g_{i2}, ..., g_{iN_i}\}$. A dialogue D_i consists of M turns of utterances between a user $\mathbf{u}$ and a system agent $\mathbf{s}$ $(\mathbf{w_{u1}}, \mathbf{w_{s1}}, \mathbf{w_{u2}}, \mathbf{w_{s2}}, ...)$, where $\mathbf{w_{u1}}$ is a word sequence $w_{u1,1}, w_{u1,2}, ..., w_{u1,N_{u1}}$ denoting the user's first utterance.

The task-oriented dialogue modeling aims to approximate the conditional probability of user's or agent's next utterance given previous turns and goals. It can be further decomposed over gener-

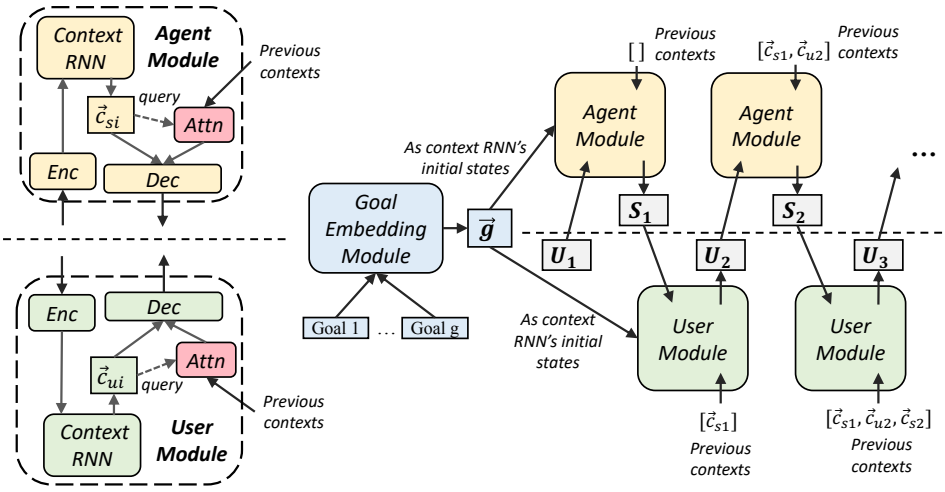


Figure 2: G-DuHA architecture. The goal embedding module embeds goals as priors for context RNNs. Dual hierarchical RNNs naturally model two interlocutors. An attention over previous contexts captures long-term dependencies. For encoders, word attentions are used to summarize local importance of words. (Enc: Encoder, Dec: Decoder, Attn: Attention, U_i: User's utterance, S_i: Agent's utterance)

ated words, e.g.

$$P(\mathbf{w_{um}}|\mathbf{w_{u1}}, \mathbf{w_{s1}}, ..., \mathbf{w_{s(m-1)}}, \mathbf{g_i}) = \prod_{n=1}^{N_{um}} P(w_{um,n}|w_{um,<n}, \mathbf{w_{u1}}, ...) \quad (1)$$

To model goal-oriented dialogues between two interlocutors, we propose Goal-embedded Dual Hierarchical Attentional Encoder-Decoder (G-DuHA) as illustrated in Fig. 2. Our model comprises goal embedding module, dual hierarchical RNNs, and attention mechanisms detailed below.

3.1 Goal Embedding Module

We represent each goal in $\{g_{i1}, g_{i2}, ...\}$ using a simple and straightforward binary or multi-one-hot encoding followed by a feed-forward network (FFN). A goal could have a specific domain such as hotel or a request such as price or area, Table 2 shows a few examples. Our goal embedding module, a FFN, then converts binary encoding of each goal into a goal embedding, where the FFN is learned during training. If multiple goals present, all goal embeddings are then added up element-wisely to be the final goal embedding:

$$\overrightarrow{g_i} = \sum_{j=1}^{|g_i|} FFN(Encode(g_{ij})) \quad (2)$$

The output of the goal embedding module has the same number of dimensions as context RNN's hidden state and is used as initial states for all layers of all context RNNs to inform the model about a set of goals to focus on.

3.2 Dual Hierarchical Architecture

The hierarchical encoder-decoder structure (Serban et al., 2016b) is designed for utterance level and context level modeling. With a single encoder, context RNN, and decoder, the same module is used to process input utterances, track contexts, and generate responses for both interlocutors.

In task-oriented dialogues, however, roles are distinct as the user aims to request information or make reservations to achieve goals in mind and the system agent provides necessary help. To model this interlocutor-level disparity, we extend it into a dual architecture involving two hierarchical RNNs, each serves as a role in a dialogue.

3.3 Attention Mechanisms

At the utterance level, as importance of words can be context dependent, our model uses first hidden layer's states of the context RNN as the query for attention mechanisms (Bahdanau et al., 2015; Xu et al., 2015) to build an utterance representation. A feed-forward network is involved to computed attention scores, whose input is the concatenation of the query and an encoder output.

At the dialogue level, for faster training, our model has a skip connection to add up context RNN's raw output with its input as the final output c_t. To model long-term dependencies, an attention module is applied to summarize all previous

contexts into a global context vector. Specifically, a feed-forward network takes the current context RNN's output c_t as the query and all previous context outputs from both context RNNs as keys and values to compute attention scores. The global context vector is then concatenated with c_t to form our final context vector for decoder to consume.

3.4 Objective

For predicting the end of a dialogue, we exploit a feed-forward network over the final context vector for a binary prediction. Thus our training objective can be written as

$$L = \sum_{i=1}^{K} \Big[-\log p_{\boldsymbol{\theta}}(D_i) + \sum_{t}^{M_i} -\log p_{\boldsymbol{\theta}}^{end}(e_{it}) \Big], \tag{3}$$

where our model $p_{\boldsymbol{\theta}}$ has parameters $\boldsymbol{\theta}$, M_i is the number of turns, e_{it} is 0 as the dialogue D_i continues and 1 if it terminates at turn t.

3.5 Generation

At dialogue generation time, a set of goals $\{g_{i1}, g_{i2}, ...\}$ and a user utterance $\mathbf{w_{u1}}$ as a seed are given. Then our model will generate conversations simulating interactions between a user and an agent that seek to complete all given goals. The generation process terminates as the end of dialogue prediction outputs a positive or the maximum number of turns is reached.

4 Experiments

We evaluate our approach on dialogue generation and response generation as well as by humans. Ablation studies and an extrinsic evaluation that leverages dialogue generation as a data augmentation method are reported in the subsequent section.

4.1 Dataset

Experiments are conducted on a task-oriented human-human written conversation dataset called MultiWOZ (Budzianowski et al., 2018), the largest publicly available dataset in the field. Dialogues in the dataset span over diverse topics, one to more goals, and multiple domains such as `restaurant`, `hotel`, `train`, etc. It consists of 8423 train dialogues, 1000 validation and 1000 test dialogues with on average 15 turns per dialogue and 14 tokens per turn.

4.2 Baselines

We compare our approach against four baselines:

(i) LM+G: As long-established methods for language generation, we adopt an RNN language model (LM) with 3-layer 200-hidden-unit GRU (Cho et al., 2014) incorporating our goal embedding module as a baseline, which has goal information but no explicit architecture for dialogues.

(ii) LM+G-XL: To show the possible impact of model size, a larger LM that has a 3-layer 450-hidden-unit GRU is adopted as another baseline.

(iii) Hierarchical recurrent encoder-decoder (HRED) (Serban et al., 2016b): As the prominent model for dialogues, we use HRED as the baseline that has a dialogue-specific architecture but no goal information. The encoder, decoder, and context RNN are 2-layer 200-hidden-unit GRUs.

(iv) HRED-XL: We also use a larger HRED with 350 hidden units for all GRUs as a baseline to show the impact of model size.

4.3 Implementation Details

In all experiments, we adopt the delexicalized form of dialogues as shown in Table 2 with vocabulary size, including slots and special tokens, to be 4258. The max number of turns and sequence length are capped to 22 and 36, respectively.

G-DuHA uses 2-layer, 200-hidden-unit GRUs as all encoders, decoders, and context RNNs. All feed-forward networks have 2 layers with non-linearity. FFNs of encoder attention and end of dialogue prediction have 50 hidden units. The FFNs of context attention and goal embedding gets 100 and 200 hidden units. We simply use greedy decoding for utterance generation.

All models initialize embeddings with pre-trained fast-text vectors on wiki-news (2018) and are trained by the Adam optimizer (2015) with early-stopping to prevent overfitting. To mitigate the discrepancy between training and inference, we pick predicted or ground-truth utterance as the current input uniformly at random when training.

4.4 Evaluation Metrics

We employ a number of automatic metrics as well as human evaluations to benchmark competing models on quality, diversity, and goal focus:

Quality. BLEU (Papineni et al., 2002), as BLEU-4 by default, is a word-overlap measure against references and commonly used by dialogue generation works to evaluate quality (2015;

Model	Size	BLEU	B1	B2	B3	D-1	D-2	D-U	P	R	F1	L-D	L-U
LM+G	4.2 M	6.34	23.24	12.89	8.76	0.16	0.88	23.75	89.52	82.39	**84.89**	15.1	13.0
LM+G-XL	8.2 M	6.22	23.10	12.75	8.63	0.16	0.93	**26.38**	**90.03**	81.62	84.71	14.7	14.4
HRED	5.1 M	5.40	21.91	11.58	7.66	0.09	0.38	3.94	69.69	66.22	65.35	17.3	14.1
HRED-XL	8.8 M	5.08	20.45	10.90	7.22	0.11	0.50	5.51	68.25	65.93	65.09	15.9	16.9
G-DuHA	7.8 M	**7.48**	**24.38**	**14.44**	**10.12**	**0.17**	**0.99**	**25.98**	86.21	**83.18**	83.76	14.3	15.0

Table 1: **Dialogue** generation performance. All results are averaged over 5 random seeds and reported in percentage (%). **Size**: Total number of model parameters, **P**: Precision, **R**: Recall, **L-D**: Average dialogue length, **L-U**: Average utterance length.

Goals	LM + Goal	HRED	G-DuHA
Domain: attraction User: [name] Reqt: [**entrance fee**] Book: False Domain: restaurant User: [area, food] Reqt: [] **Book: True** Domain: **taxi** User: [arriveBy] Reqt: [car type, **phone**] Book: False	U: what is the address for [attraction_name] ? S: sure ! [attraction_name] is located at [attraction_address] and **is free .** U: i . i am also **looking for a place to eat .** the [value_area] . S: there are [value_count] restaurant -s in the [value_area] . what type of food are you looking for ? ... U: yes , **please book a table** for [value_count] people at [value_time] on [value_day] . ... **U: i help you with anything else ?** S: i **also need a taxi** to get me the restaurant to the restaurant . U: i time would you like to leave ? S: i would like to arrive by the restaurant by the booked time . U: booking completed ! booked car type [taxi_type] **contact number [taxi_phone]** S: thank you that s all i needed . U: you are welcome . have a great day !	U: what is the address for [attraction_name] ? S: the address is [hospital_address] , [value_place] . is there anything else i can help you with ? U: yes , i would like to book a table for [value_count] at [value_time] on [value_day] . S: i **have booked your table .** your reference number is [restaurant_reference] . U: thank you . i also need **a train** to [value_place] that leaves after [value_time] . S: there are [value_count] trains that match your request . what day and time would you like to travel ? U: i would like to leave on [value_day] and arrive by [value_time] . S: there are [value_count] trains that meet your criteria . would you like to narrow it down by departure time or arrival time ? U: i would like to arrive by [value_time]	U: what is the address for [attraction_name] ? S: the address is [attraction_address] and the postcode is [attraction_postcode] . U: **what is the entrance fee ?** S: the **entrance fee is free** . is there anything else i can help you with ? U: yes , i am also **looking for a restaurant** that serves [value_food] food in the [value_area] . ... S: i have [value_count] restaurant -s that meet your criteria . would you like me to **book you a table ?** **U: yes , please .** i need a table for [value_count] people at [value_time] on [value_day] . ... U: i **also need a taxi** to commute between the [value_count] place -s . S: i have booked you a taxi to pick you up at the [attraction_name] . it will be a [taxi_type] and the **contact number is [taxi_phone]** . U: thank you , that is all i need . S: you are welcome . have a great day !

Table 2: Dialogue qualitative comparison. Reqt: Requests. U: User, S: Agent. **Goal hit or miss**. **Role confusion**. Extensive qualitative comparisons of dialogues are presented in the appendix.

2016b; 2016a; 2017; 2018). Lower N-gram B1, B2, B3 are also reported.

Diversity. D-1, D-2, D-U: The distinctiveness denotes the number of unique unigrams, bigrams, and utterances normalized by each total count (Li et al., 2016a; Xu et al., 2018). These metrics are commonly used to evaluate the dialogue diversity.

Goal Focus. A set of slots such as address are extracted from reference dialogues as multi-label targets. Generated slots in model's output dialogues are the predictions. We use the multi-label precision, recall, and F1-score as surrogates to measure the goal focus and achievement.

Human Evaluation. The side-by-side human preference study evaluates dialogues on *goal focus*, *grammar*, *natural flow*, and *non-redundancy*.

5 Results and Discussion

5.1 Dialogue Generation Results

For dialogue generation (Li et al., 2016b), a model is given one or more goals and one user utterance as the seed inputs to generate entire dialogues in an auto-regressive manner.

Table 1 summarizes the evaluation results. For quality measures, G-DuHA significantly outperforms other baselines, implying that it's able to carry out a higher-quality dialogue. Besides, goal-embedded LMs perform better than HREDs, showing the benefits of our goal embedding module. No significant performance difference is observed with respect to model size variants.

For diversity evaluations, G-DuHA is on par with goal-embedded LMs and both outperform HRED significantly. Of 1000 generated dialogues, HRED delivers highly repetitive outputs with only 4 to 6% distinct utterances, whereas 25% of utterances are unique from G-DuHA.

For recovering slots in reference dialogues, precision denotes a degree of goal deviation, recall entails the achievement of goals, and F1 measures the overall focus. Goal-embedded LM is the best on precision and F1 with G-DuHA having com-

Model	BLEU	B1	B2	B3	D-1	D-2	D-U	P	R	F1	L-R
LM+G	14.88	35.86	24.59	18.81	0.27	1.44	40.84	79.71	68.57	71.73	14.3
LM+G-XL	14.51	35.28	24.07	18.36	**0.28**	**1.47**	**42.56**	**79.79**	67.31	71.00	14.3
HRED	14.34	36.27	24.31	18.33	0.21	0.94	20.21	75.46	67.08	68.78	17.1
HRED-XL	14.33	36.36	24.37	18.35	0.23	1.12	26.63	72.69	68.24	68.20	17.3
G-DuHA	**15.85**	**37.99**	**26.14**	**20.01**	0.25	1.27	**39.59**	**78.34**	**71.55**	**72.69**	16.7

Table 3: **Agent's response** generation performance. All results are averaged over 5 random seeds and reported in percentage (%). **P**: Precision, **R**: Recall, **L-R**: Average response length.

Model	BLEU	B1	B2	B3	D-1	D-2	D-U	P	R	F1	L-R
LM+G	11.73	31.79	21.26	15.56	0.35	1.82	33.57	89.44	75.78	80.23	10.6
LM+G-XL	11.60	31.49	21.00	15.38	**0.36**	**1.87**	34.29	89.57	75.55	80.03	10.7
HRED	10.88	31.69	20.46	14.65	0.24	0.98	16.00	80.00	79.11	77.58	13.1
HRED-XL	10.81	31.84	20.48	14.60	0.26	1.15	19.87	80.11	78.82	77.42	13.2
G-DuHA	**13.25**	**35.20**	**23.89**	**17.56**	0.30	1.49	**35.57**	**91.12**	**79.66**	**83.51**	12.8

Table 4: **User's response** generation performance. All results are averaged over 5 random seeds and reported in percentage (%). **P**: Precision, **R**: Recall, **L-R**: Average response length.

Goals	Contexts (Reference)	Response
Domain: train User: [destination, departure, arriveBy, day] Reqt:[duration,price] Book: False Domain: restaurant User: [area, food, pricerange] Reqt: [] Book: True	U: hey i am looking for a train from [value_place] to [value_place] . mind helping out ? S: many trains are leaving on [value_day] . mind choosing the time ? U: i want to leave on [value_day] and arrive by [value_time] . S: i have train [train_id] that leaves [value_place] at [value_time] and arrives in [value_place] at [value_time] . would you like to make reservations ? U: could you give me the travel time and price of that train please ? S: the cost is [value_price] pounds . the duration of the trip is [value_count] minutes , leaving [value_place] at [value_time] and arriving at [value_time] in [value_place] .	**LM+G**: thanks . i am also looking for a restaurant in the [value_area] . **HRED**: i am also looking for a place to stay . i would like a [value_count] star hotel . **G-DuHA**: thank you . i am also looking for a restaurant in the [value_area] of town .
Domain: train User: [destination, departure, arriveBy, day] Reqt: [] Book: False	U: i need a train that leaves on [value_day] with an arrival of [value_time] . S: where will you be travelling to and from ?	**LM+G**: i will be departing from [value_place] . **HRED**: i will be leaving from [value_place] . **G-DuHA**: i am departing from [value_place] and need to arrive by [value_time] .

Table 5: Qualitative comparison of generated responses. Reqt: Requests. U: User, S: Agent.

parable performance. However, even thought LM can better mention the slots in dialogue generation, utterances are often associated with a wrong role. That is, role confusions are commonly seen such as the user makes reservations for the agent as in Table 2. The reason could be that LM handles the task similar to paragraph generation without an explicit design for the conversational hierarchy.

Overall, G-DuHA is able to generate high-quality dialogues with sufficient diversity and still adhere to goals compared to baselines.

Qualitative Comparison. Table 2 compares generated dialogues from different models given one to three goals to focus on. It's clear that models with the goal embedding module are able to adhere to given goals such as "book" or "no book", requesting "price" or "entrance fee" while HRED fails to do so. They can also correctly covering all required domain contexts without any diversion such as switching from attraction inquiry to restaurant booking, then to taxi-calling. For HRED, without goals, generated dialogues often detour to a non-relevant domain context such as shifting to train booking while only hotel inquiry is required.

For goal-embedded LM, a serious issue revealed is role confusions as LM often wrongly shifts between the user and agent as shown in Table 2. The issue results from one wrong `EndofUtterance` prediction but affects rest of dialogue and degrades the overall quality. More generated dialogues are reported in the appendix.

5.2 Response Generation Results

For response generation (Sordoni et al., 2015; Serban et al., 2016c; Park et al., 2018), a set of goals as well as the previous context, i.e. all previous

	Wins	Losses	Ties
Goal Focus	**82.33%**	6.00%	11.67%
Grammar	6.00%	5.00%	89.00%
Natural Flow	**26.00%**	15.00%	59.00%
Non-redundancy	**35.34%**	6.33%	58.33%

Table 6: Human evaluations, G-DuHA vs HRED. 100 pairs of generated dialogues along with goals are given to three domain experts for side-by-side comparisons.

reference utterances, are given to a model to generate the next utterance as a response.

Table 3 and 4 summarize the results. G-DuHA outperforms others on quality and goal focus measures and rivals LM-goal on diversity on both agent and user responses. For goal focus, LM-goal performs good on precision but short on recall. This could because it generates much shorter user and agent responses on average.

Interestingly, as previous contexts are given, LM-goal performs only slightly better than HRED. This implies hierarchical structures capturing longer dependencies can make up the disadvantages of having no goal information for response generation. However, as illustrated in Table 12, HRED could still fail to predict the switch of domain contexts, e.g. from `train` to `restaurant`, which explains performance gaps. Another intriguing observation is that when incorporating the goal embedding module, response diversity and goal focus can be boosted significantly.

Comparing the performance between agent and user response generation, we observe that models can achieve higher quality and diversity but lower goal focus when modeling agent's responses. These might result from the relatively consistent utterance patterns but diverse slot types used by an agent. More generated responses across different models are presented in the appendix.

5.3 Human Evaluation Results

For human evaluation, we conduct side-by-side comparisons between G-DuHA and HRED, the widely used baseline in literature, on dialogue generation task. We consider the following four criteria: *goal focus*, *grammar*, *natural flow*, and *non-redundancy*. *Goal focus* evaluates whether the dialogue is closely related to the preset goals; *grammar* evaluates whether the utterances are well-formed and understandable; *natural flow* evaluates whether the flow of dialogue is logical and fluent; and *non-redundancy* evaluates whether the dialogue is absent of unnecessary repetition of mentioned information. 100 pairs of generated dialogues from G-DuHA and HRED along with their goals are randomly placed against each other. For each goal and pair of dialogues, three domain experts were instructed to set their preferences with respect to each of the four criteria, marked as *win / lose / tie* between the dialogues.

Table 6 presents the results. G-DuHA shows substantial advantages on goal focus, with 82.33% wins over HRED, confirming the benefits of our goal embedding module. G-DuHA also outperforms HRED significantly on natural flow and non-redundancy. These might result from G-DuHA's ability to generating much more diverse utterances while concentrating on current goals. An especially interesting observation is that in cases where multiple goals are given, G-DuHA not only stays focused on each individual goal but also generates intuitive transitions between goals, so that the flow of a dialogue is natural and coherent. An example is shown in Table 2, where the G-DuHA-generated dialogue switches towards the `taxi` goal while maintaining reference to the previously mentioned `attraction` and `restaurant` goals: *"... i also need a taxi to commute between the 2 places ..."*. We also observe that both G-DuHA and HRED performed well on grammaticality. The generated samples across all RNN-based models are almost free from grammar error as well.

5.4 Ablation Studies

The ablation studies are reported in Table 7 for dialogue generation and in Table 8 for response generation to investigate the contribution of each module. Here we evaluate user and agent response generation together.

Goal Embedding Module. First, we examine the impact of goal embedding module. When unplugging the goal embedding module, we observe significant and consistent drops on quality, diversity, and goal focus measures for both dialogue and response generation tasks. For dialogue generation task, the drops are substantially large which resonates with our intuition as the model only has the first user utterance as input context to follow.

With no guideline about what to achieve and what conversation flow to go around with, dialogues generated from HRED often have the similar flow and low diversity. These results demon-

Model	BLEU	B1	D-1	D-2	D-U	P	R	F1
G-DuHA	7.48	24.38	0.17	0.99	25.98	86.21	83.18	83.76
w/o goal	5.19	20.04	0.13	0.68	13.83	69.18	68.21	66.86
w/o dual	7.34	24.99	0.15	0.79	19.22	85.24	82.62	82.96
w/o context attention	7.34	24.34	0.17	0.99	24.98	86.70	83.40	84.10

Table 7: Ablation studies on **dialogue** generation over goal embedding module, dual architecture, and dialogue-level attention. Results are averaged over 5 random seeds and reported in percentage (%). **P**: Precision, **R**: Recall.

Model	BLEU	B1	D-1	D-2	D-U	P	R	F1
G-DuHA	14.84	36.84	0.18	1.10	34.23	89.87	82.00	84.80
w/o goal	13.29	35.23	0.16	0.97	28.04	84.33	80.15	81.10
w/o dual	14.60	36.66	0.17	0.96	27.53	89.43	80.81	83.91
w/o context attention	14.73	36.88	0.18	1.14	34.55	90.28	81.54	84.80

Table 8: Ablation studies on **response** generation over goal embedding module, dual architecture, and dialogue-level attention. Results are averaged over 5 random seeds and reported in percentage (%). **P**: Precision, **R**: Recall.

	Joint Goal	Turn Request
GLAD	88.55%	97.11%
GLAD + LM+G	88.07%	96.02%
GLAD + HRED	89.03%	97.11%
GLAD + G-DuHA	**89.04%**	**97.59%***

Table 9: Test accuracy of GLAD (Zhong et al., 2018) on the WoZ restaurant reservation dataset with different data augmentation models. (*significant against others.)

strate that our goal embedding module is critical in generating higher-quality and goal-centric dialogues with much more diversity.

Dual Hierarchical Architecture. We also evaluate the impact of dual hierarchical architecture. Comparisons on both dialogue and response generation tasks show a consistent trend. We observe that applying dual architecture for interlocutor-level modeling leads to a solid increase in utterance diversity as well as moderate improvements on quality and goal focus.

The results echo our motivation as two interlocutors in a goal-oriented dialogue scenario exhibit distinct conversational patterns and this interlocutor-level disparity should be modeled by separate hierarchical encoder-decoders.

For the dialogue-level attention module, there is no significant effect on diversity and goal focus on both tasks but it marginally improves the overall utterance quality as BLEU scores go up by a bit.

6 Data Augmentation via Dialogue Generation

As an exemplified extrinsic evaluation, we leverage the goal-oriented dialogue generation as data augmentation for task-oriented dialogue systems. Dialogue state tracking (DST) is used as our evaluation task which is a critical component in task-oriented dialogue systems (Young et al., 2013) and has been studied extensively (Henderson et al., 2014; Mrksic et al., 2017; Zhong et al., 2018).

In DST, given the current utterance and dialogue history, a dialogue state tracker determines the state of the dialogue which comprises a set of *requests* and *joint goals*. For each user turn, the user informs the system a set of turn goals to fulfill, *e.g. inform(area=south)*, or turn requests asking for more information, *e.g. request(phone)*. The joint goal is the collection of all turn goals up to the current turn.

We use the state-of-the-art Global-Locally Self-Attentive Dialogue State Tracker (GLAD) (Zhong et al., 2018) as our benchmark model and the WoZ restaurant reservation dataset (Wen et al., 2017; Zhong et al., 2018) as our benchmark dataset, which is commonly used for the DST task.

The dataset consists of 600 train, 200 validation and 400 test dialogues. We use the first utterances from 300 train dialogues and sample restaurant-domain goals to generate dialogues, whose states are annotated by a rule-based method.

Table 9 summarizes the augmentation results. Augmentation with G-DuHA achieved an improvement over the vanilla dataset and outperform

HRED on turn requests while being comparable on joint goal. For goal-embedded LM, as it struggles with role confusion, the augmentation actually hurts the overall performance.

7 Conclusion

We introduced the goal-embedded dual hierarchical attentional encoder-decoder (G-DuHA) for goal-oriented dialogue generation. G-DuHA is able to generate higher-quality and goal-focused dialogues as well as responses with decent diversity and non-redundancy. Empirical results show that the goal embedding module plays a vital role in the performance improvement and the dual architecture can significantly enhance diversity.

We demonstrated one application of the goal-oriented dialogue generation through a data augmentation experiment, though the proposed model is applicable to other conversational AI tasks which remains to be investigated in the future.

As shown in experiments, a language model coupled with goal embedding suffers from role-switching or confusion. It's also interesting to further dive deep with visualizations (Kessler, 2017) and quantify the impact on quality, diversity, and goal focus metrics.

Acknowledgments

The authors would like to acknowledge the entire AWS Lex Science team for thoughtful discussions, honest feedback, and full support. We are also very grateful to the reviewers for insightful comments and helpful suggestions.

References

Nabiha Asghar, Pascal Poupart, Jesse Hoey, Xin Jiang, and Lili Mou. 2018. Affective neural response generation. In *European Conference on Information Retrieval*, pages 154–166. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*.

Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, and Dawei Yin. 2018. Hierarchical variational memory network for dialogue generation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1653–1662. International World Wide Web Conferences Steering Committee.

Yun-Nung Chen, Dilek Hakkani-Tür, Gökhan Tür, Jianfeng Gao, and Li Deng. 2016. End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Interspeech*, pages 3245–3249.

Kyunghyun Cho, Bart van Merrienboer, aglar Gülehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*.

Milica Gašić and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.

Joshua T Goodman. 2001. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.

Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tr. 2019. Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations. In *Proc. Interspeech 2019*, pages 1891–1895.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017. Improving neural language models with a continuous cache. *CoRR*, abs/1612.04426.

Arshit Gupta, John Hewitt, and Katrin Kirchhoff. 2019. Simple, fast, accurate intent classification and slot labeling. *arXiv preprint arXiv:1903.08268*.

Matthew Henderson, Paweł Budzianowski, Iñigo Casanueva, Sam Coope, Daniela Gerz, Girish Kumar, Nikola Mrkšić, Georgios Spithourakis, Pei-Hao Su, Ivan Vulic, and Tsung-Hsien Wen. 2019. A repository of conversational datasets. In *Proceedings of the Workshop on NLP for Conversational AI*. Data available at github.com/PolyAI-LDN/conversational-datasets.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing*, 35(3):400–401.

Jason S. Kessler. 2017. Scattertext: a browser-based tool for visualizing how corpora differ. In *Proceedings of ACL-2017 System Demonstrations*, Vancouver, Canada. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *1995 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184. IEEE.

Hung Le, Truyen Tran, Thin Nguyen, and Svetha Venkatesh. 2018a. Variational memory encoder-decoder. In *Advances in Neural Information Processing Systems*, pages 1508–1518.

Hung Le, Truyen Tran, Thin Nguyen, and Svetha Venkatesh. 2018b. Variational memory encoder-decoder. In *NeurIPS*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B. Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *HLT-NAACL*.

Jiwei Li, Will Monroe, Alan Ritter, Daniel Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *EMNLP*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Daniel Jurafsky. 2017a. Adversarial learning for neural dialogue generation. In *EMNLP*.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli elikyilmaz. 2017b. End-to-end task-completion neural dialogue systems. In *IJCNLP*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2017. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*.

Gábor Melis, Chris Dyer, and Phil Blunsom. 2018. On the state of the art of evaluation in neural language models. *CoRR*, abs/1707.05589.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing lstm language models. *CoRR*, abs/1708.02182.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Z. Hakkani-Tür, Xiaodong He, Larry P. Heck, Gökhan Tür, Dong Yu, and Geoffrey Zweig. 2015. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23:530–539.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. IEEE.

Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Yookoon Park, Jaemin Cho, and Gunhee Kim. 2018. A hierarchical latent structure for variational conversation modeling. In *NAACL-HLT*.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. 2005. Let's go public! taking a spoken dialog system to the real world. In *Ninth European conference on speech communication and technology*.

Iulian Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C. Courville. 2016a. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016b. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2016c. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and William B. Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *HLT-NAACL*.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina Maria Rojas-Barahona, Pei hao Su, Stefan Ultes, David Vandyke, and Steve J. Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan R. Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*.

Xinnuo Xu, Ondrej Dusek, Ioannis Konstas, and Verena Rieser. 2018. Better conversations by modeling, filtering, and optimizing for coherence and diversity. In *EMNLP*.

Kang Min Yoo, Youhyun Shin, and Sang goo Lee. 2019. Data augmentation for spoken language understanding via joint variational generation. *CoRR*, abs/1809.02305.

Steve J. Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101:1160–1179.

Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *ACL*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. *CoRR*, abs/1805.09655.

A Appendix

A.1 More Generated Dialogues for Qualitative Comparison

In Table 10 and Table 11, we list and compare more generated dialogues of different models given a variable number of goals to focus on.

The domains, what slots to request such as phone or postcode, and when should the user make a reservation or not are specified in goals. We use bold style to emphasize whether the assigned goals are achieved or missed such as requests of a correct slot or a wrong domain. The underline and bold style denotes the role confusions.

A.2 More Generated Responses for Qualitative Comparison

Responses are generated by a model given the goals and all the previous utterances. In Table 12, we present more generated responses for qualitative comparison. We observe that HRED is unable to correctly switch beteen goal contexts illustrated by the second example.

Goals	LM + Goal	HRED	G-DuHA
Domain: **hotel** User: [name] Reqt: [] **Book: True**	U: i am looking for a hotel called the [hotel_name] . S: [hotel_name] is a [value_pricerange] -ly priced guesthouse in the [value_area] . would you like me to book a room for you ? U: yes , please . i need [value_count] for [value_count] people and [value_count] nights starting from [value_day] . S: **i have booked you at the [hotel_name] .** your reference number is [hotel_reference] . is there anything else i can help you with ? U: no , that s all . thank you . S: you are welcome . have a great day !	U: i am looking for a hotel called the [hotel_name] . S: [hotel_name] is a guesthouse in the [value_area] . it is [value_pricerange] -ly priced and has [value_count] stars . would you like to book a room ? U: yes , please . i need a room for [value_count] people for [value_count] nights starting on [value_day] . S: i am sorry , but i was unable to book that for you . would you like to try a different day or a shorter stay ? U: how about [value_count] nights ? S: **i was able to book you at the [hotel_name] .** your reference number is [hotel_reference] . is there anything else i can help you with ? U: **i am also looking for a train** that leaves after [value_time] on [value_day] . ... (Cut Off) → **Off the Rails**	U: i am looking for a hotel called the [hotel_name] . S: [hotel_name] is a guesthouse in the [value_area] area . would you like to book a room ? U: yes , i would like to book it for [value_count] people and [value_count] nights starting from [value_day] . S: **booking was successful .** reference number is [hotel_reference] . U: that s all i need . thank you . S: you are welcome . have a great day !
Domain: attraction User: [name] Reqt: [**phone**, type] Book: False Domain: train User: [arriveBy, day, departure, destination] Reqt: [**price**] **Book: False**	U: i am looking for information about [attraction_name] S: [attraction_name] is a nightclub in the [value_area] of town . their **phone number is [attraction_phone]** . **U: i help you with anything else ?** S: yes , i am also looking for a train that [value_place] to [value_place] . U: what day and time would you like to leave ? S: i would like to leave on [value_day] and arrive by [value_time] . U: i have [value_count] trains that that day . [value_time] . would you like me to book it for you ? S: yes , please . **i need the price** and train id . U: the train is [value_price] pounds . the price is [value_price] pounds . S: you like me to **book that for you ?** **U: no** , that s all i need . thank you . S: thank you for using our service . have a great day .	U: i am looking for information about [attraction_name] S: [attraction_name] is located in the [value_area] of town at [attraction_address] . U: what is the entrance fee ? S: i am sorry , i do not have that information . U: i am also looking for a train that leaves after [value_time] on [value_day] . S: i have [value_count] trains that meet your criteria . would you like to narrow it down by departure site , destination , or arrival time ? U: i would like to leave on [value_day] and arrive by [value_time] . S: i have train [train_id] that leaves at [value_time] and arrives at [value_time] . would you like me to book that for you ? U: **yes , please book it** for [value_count] people . S: booking was successful , the total fee is [value_price] gbp payable at the station . reference number is [train_reference] . U: thank you . that is all i need . S: you are welcome . have a great day !	U: i am looking for information about [attraction_name] S: [attraction_name] is a museum in the [value_area] area . would you like the phone number ? U: yes , i would like the postcode and phone number . S: their **phone number is [attraction_phone]** and the postcode is [attraction_postcode] . U: i also need a train to [value_place] . S: where will you be departing from ? U: i will be leaving from [value_place] on [value_day] . S: i have [value_count] trains that meet your criteria . what time would you like to arrive by ? U: i need to arrive by [value_time] . S: i have train [train_id] that leaves at [value_time] and arrives at [value_time] . would you like me to **book that for you ?** **U: no** , that s all i needed . thank you . S: you are welcome . have a great day !

Table 10: Dialogue qualitative comparison. Reqt: Requests. U: User, S: System. **Goal hit or miss**. **Role confusion**.

Goals	LM + Goal	HRED	G-DuHA
Domain: attraction User: [name] Reqt: [address, entrance fee] Book: False Domain: **hotel** User: [name] Reqt: [postcode, stars] **Book: False** Domain: **taxi** User: [leaveAt] Reqt: [car type, phone] Book: False	U: i am looking for [attraction_name] in [value_place] S: it is located in the [value_area] area . it is located on [attraction_address] . **U: is free .** **S: can i get the phone number ?** U: the , the phone number is [attraction_phone] . S: i you . **i am also looking for a hotel called [hotel_name] .** U: [hotel_name] is a [value_pricerange] hotel in the [value_area] area . it is a [value_count] star rating . would you like me to **book a room for you ?** **S: no ,** i just need the information . U: you are welcome . is there anything else i can help you with today ? S: **i also need a taxi** to commute between the [value_count] place -s . U: what time would you like to leave ? S: i would like to leave the hotel by [value_time] . U: i have booked a taxi for you . the contact number is [taxi_phone] . is there anything else i can help you with ? S: no , that s all i need . thank you for your help ! U: you are welcome . have a great day !	U: i am looking for [attraction_name] in [value_place] S: i have found [attraction_name] , it s located in the [value_area] . it is free to enter . would you like more information ? U: yes , can i get the phone number and postcode ? S: sure , the phone number is [attraction_phone] and the postcode is [attraction_postcode] . **U: i am also looking for a train** that leaves on [value_day] and goes to [value_place] . S: there are [value_count] trains that match your request . what time would you like to leave ? U: i would like to leave after [value_time] . S: i have train [train_id] that leaves at [value_time] and arrives at [value_time] . would you like me to book that for you ? U: yes , please book it for [value_count] people . S: booking was successful , the total fee is [value_price] gbp payable at the station . reference number is [train_reference] . U: thank you so much for your help . S: you are welcome . have a great day ! U: thank you , you too . S: thank you for using our service today ! → **Off the Rails**	U: i am looking for [attraction_name] in [value_place] S: it is located at [attraction_address] . is there anything else you would like to know ? U: what is the postcode ? S: the postcode is [hotel_postcode] . is there anything else i can help you with ? **U: i am also looking for a hotel called [hotel_name] .** S: [hotel_name] is a [value_pricerange] -ly priced guesthouse in the [value_area] area . would you like me to **book you a room ?** **U: no ,** i just need the address please . S: the postcode is [hotel_postcode] . is there anything else i can help you with ? **U: i also need a taxi** to get between the [value_count] place -s . S: i can help you with that . what time would you like to leave or arrive by ? U: i want to leave the hotel by [value_time] . S: i have booked you a [taxi_type] , the contact number is [taxi_phone] . U: thank you so much for your help . S: you are welcome . is there anything else i can help you with today ? U: no , that s all i need . thank you for your help ! S: you are welcome . have a great day .
Domain: hotel User: [name] Reqt: [address] **Book: False** Domain: **attraction** User: [area] Reqt: [address, phone, type] Book: False	U: hello ! i am looking for information about [hotel_name] . it s a hotel in [value_place] . S: the is a guesthouse in the [value_area] part of town . it is [value_pricerange] -ly priced and has [value_count] stars . would you like me to book a room ? **U: no , i do not need a reservation .** i you . S: you are welcome . is there anything else i can help you with today ? U: yes , i am **looking for a place called [attraction_name] .** S: [attraction_name] is a museum in the [value_area] area . it is located at [attraction_address] . their phone number is [attraction_phone] . **U: i help you with anything else ?** S: no , that s all i need . thank you . U: you are welcome . have a great day !	U: hello ! i am looking for information about [hotel_name] . it s a hotel in [value_place] . S: [hotel_name] is a guesthouse in the [value_area] . it is [value_pricerange] -ly priced and has [value_count] stars . would you like to book a room ? U: yes , please . i need a room for [value_count] people for [value_count] nights starting on [value_day] . S: i am sorry , but i was unable to book that for you . would you like to try a different day or shorter stay ? U: how about [value_count] nights ? **S: i was able to book you** at the [hotel_name] . your reference number is [hotel_reference] . is there anything else i can help you with ? **U: i am also looking for a train** that leaves after [value_time] on [value_day] . ... (Cut Off) → **Off the Rails**	U: hello ! i am looking for information about [hotel_name] . it s a hotel in [value_place] . S: [hotel_name] is a guesthouse in the [value_area] area . it is a [value_count] star hotel in the [value_area] area . would you like me to book a room for you ? **U: no , i just need the address** and phone number . S: sure , the phone number is [attraction_phone] . **U: i am also looking for a place to go in town** . S: there are [value_count] attractions in the [value_area] area . do you have a particular type of attraction in mind ? U: i am not sure . i just need the address and postcode for [value_count] of them . S: [attraction_name] is located at [attraction_address] and the postcode is [attraction_postcode] . U: no , that s all i need . thank you for your help ! S: you are welcome . have a great day !
Domain: train User: [day, departure, destination, leaveAt] Reqt: [**duration, price**] **Book: False**	U: i am looking for a train from [value_place] to [value_place] . S: there are [value_count] trains that meet your criteria . what day and time would you like to travel ? U: i would like to leave on [value_day] after [value_time] . S: i have [value_count] trains that leave [value_place] at [value_time] . would you like me to **book it for you ?** **U: yes ,** please . i need **the price and travel time please .** S: the price is [value_price] pounds . the price is [value_price] pounds . **U: you like me to book that for you ?** S: no , that s all i need . thank you . U: you are welcome . have a great day !	U: i am looking for a train from [value_place] to [value_place] . S: there are [value_count] trains that meet your criteria . what day and time would you like to travel ? U: i would like to leave on [value_day] and arrive by [value_time] . S: i have train [train_id] that leaves at [value_time] and arrives in [value_place] at [value_time] . would you like me to **book that for you ?** **U: yes ,** please . i need [value_count] tickets . S: i have booked [value_count] tickets for you . your reference number is [train_reference] . U: i am also looking for a place to stay . **i would like a [value_count] star hotel .** ... (Cut Off) → **Off the Rails**	U: i am looking for a train from [value_place] to [value_place] . S: what day would you like to travel ? U: i would like to leave on [value_day] . S: there are [value_count] trains that meet your criteria . do you have a time you would like to leave or arrive by ? U: i would like to leave after [value_time] . S: [train_id] leaves at [value_time] and arrives at [value_time] . would you like me to **book it for you ?** **U: no ,** i just need **the price** and the train id . S: the price is [value_price] pounds . U: thank you for your help . that is all i need today . S: you are welcome . have a great day !

Table 11: Dialogue qualitative comparison (continue). Reqt: Requests. U: User, S: System. **Goal hit or miss**. **Role confusion**.

Goals	Contexts (Reference)	Response
Domain: attraction User: [type] Reqt: [entrance fee, address, phone] Book: False Domain: restaurant User: [pricerange, food] Reqt: [phone, postcode] Book: False	U: can you find me a theatre ? S: would you like the [attraction_name] ? U: yes and i need to know the entrance fee , address , and phone number S: the fee is not listed but the address is [attraction_address] and the phone number is [attraction_phone] . can i help you with anything else ? U: i am also looking for a [value_food] restaurant in the [value_pricerange] range . S: i have [value_count] excellent options . [restaurant_name] is in the [value_area] of town , would you like me to reserve you a table ? U: i am not sure how many people will be coming , so i will do that later. but can you provide me with their postcode and phone number ?	**LM+G**: the phone number is [restaurant_phone] and the postcode is [restaurant_postcode] . **HRED**: the phone number is [restaurant_phone] and the postcode is [restaurant_postcode] . **G-DuHA**: sure , the phone number is [restaurant_phone] and the postcode is [restaurant_postcode] . is there anything else i can help you with ?
Domain: train User: [day, departure, destination, leaveAt] Reqt: [] Book: True Domain: attraction User: [area, type] Reqt: [phone, postcode] Book: False	U: i am looking for a train that leaves on [value_day] going to [value_place] . S: there are [value_count] trains that meet what you are asking . where would you like to depart from ? U: [value_place] , it should also leave after [value_time] . S: [train_id] departs [value_place] at [value_time] and arrives in [value_place] at [value_time] . will that work ? U: yes it will . can you book that for just me , and provide the reference number ? S: sure thing ! how many tickets do you need ? U: i need [value_count] ticket and can i please have the reference number ? S: booking was successful , the total fee is [value_price] gbp payable at the station . reference number is [train_reference] . can i provide any further help today ?	**LM+G**: i am also looking for a place to go in the [value_area] . **HRED**: i am also looking for a place to stay . i would like a hotel in the [value_area] . **G-DuHA**: i am also looking for a place to go in town .
Domain: hotel User: [area, internet, parking, stars] Reqt: [postcode, pricerange, type] Book: False Domain: restaurant User: [area, food, pricerange] Reqt: [] Book: True	U: hi i am looking for a restaurant in the [value_area] of town serving [value_food] food . S: that s great ! there are several [value_food] restaurant -s near the city [value_area] . are you looking for a budget @UNK option or are you comfortable with a more [value_pricerange] place ? U: a [value_pricerange] option please . S: we have [restaurant_name] , the ghandi and [restaurant_name] , do you want me to book you for [value_count] of those ? U: yes please for [value_count] people at [value_time] on [value_day]. S: you are all set . they will hold the table for [value_count] minutes and your reference number is [hotel_reference] U: thank you , appreciate your service . S: no problem . is there anything else i can help you with ?	**LM+G**: i am also looking for a place to stay . i would like a [value_count] star hotel in the [value_area] . **HRED**: no , that s all . thank you . **G-DuHA**: i am also looking for a place to stay .
Domain: taxi User: [destination, arriveBy, departure] Reqt: [car type, phone] Book: False	U: i need to book a tax departing from gandhi . S: what is your destination and when would you like to depart ?	**LM+G**: i need to arrive by [value_time] . **HRED**: i am going to [value_place] from [value_place] . **G-DuHA**: i want to go to [value_place] and get to [restaurant_name] by [value_time] .
Domain: attraction User: [area] Reqt: [address] Book: False Domain:hotel User: [pricerange, internet, stars] Reqt: [] Book: True	U: hello , i am looking for place -s to go in the [value_area] . S: there are many attractions in the [value_area] like museums , architecture , boating , and @UNK . what are you interested in ? U: how about a boating attraction ? S: there are [value_count] in the [value_area] of town . [hotel_name] , and [hotel_name] . would either of those interest you ? U: could you give me the address for [attraction_name] , please ? i also need a place to stay , preferably somewhere [value_pricerange] . S: sure , [attraction_name] is @UNK at [attraction_address] . the [hotel_name] is in the [value_pricerange] price range with [value_count] stars . would you like to book a room ? U: does it offer free wifi internet ?	**LM+G**: yes , it does . would you like me to book a room for you ? **HRED**: yes , it does have free parking . **G-DuHA**: yes , it does .

Table 12: Qualitative comparison of generated responses. Reqt: Requests. U: User, S: System.

Putting the Horse Before the Cart: A Generator-Evaluator Framework for Question Generation from Text

Vishwajeet Kumar[1,2,3], **Ganesh Ramakrishnan**[2], and **Yuan-Fang Li**[3]

[1]IITB-Monash Research Academy, Mumbai, India
[2]IIT Bombay, Mumbai, India
[3]Monash University, Melbourne, Australia

Abstract

Automatic question generation (QG) is a useful yet challenging task in NLP. Recent neural network-based approaches represent the state-of-the-art in this task. In this work, we attempt to strengthen them significantly by adopting a holistic and novel generator-evaluator framework that directly optimizes objectives that reward semantics and structure. The *generator* is a sequence-to-sequence model that incorporates the *structure* and *semantics* of the question being generated. The generator predicts an answer in the passage that the question can pivot on. Employing the copy and coverage mechanisms, it also acknowledges other contextually important (and possibly rare) keywords in the passage that the question needs to conform to, while not redundantly repeating words. The *evaluator* model evaluates and assigns a reward to each predicted question based on its conformity to the *structure* of ground-truth questions. We propose two novel QG-specific reward functions for text conformity and answer conformity of the generated question. The evaluator also employs structure-sensitive rewards based on evaluation measures such as BLEU, GLEU, and ROUGE-L, which are suitable for QG. In contrast, most of the previous works only optimize the cross-entropy loss, which can induce inconsistencies between training (objective) and testing (evaluation) measures. Our evaluation shows that our approach significantly outperforms state-of-the-art systems on the widely-used SQuAD benchmark as per both automatic and human evaluation.

1 Introduction

Automatic question generation (QG) is a very important yet challenging problem in NLP. It is defined as the task of generating syntactically correct, semantically sound and relevant questions from various input formats such as text, a structured database or a knowledge base (Mannem et al., 2010). More recently, neural network based techniques such as sequence-to-sequence (Seq2Seq) learning have achieved remarkable success on various NLP tasks, including question generation. A recent deep learning approach to question generation (Serban et al., 2016) investigates a simpler task of generating questions only from a triplet of subject, relation and object. Learning to ask (referred to as L2A hereinafter) (Du et al., 2017) proposes a Seq2Seq model with attention for question generation from text. (Song et al., 2018) (in an approach referred to as NQG$_{LC}$ hereafter) encoded ground-truth answers and employed bi-directional LSTMs in a Seq2Seq setting. In addition, they use the *copy* mechanism (See et al., 2017) and context matching to capture interactions between the given ground-truth answer and its context within the passage.

In the context of QG from paragraphs, (Zhao et al., 2018) proposed maxout pointer network to keep track of word *coverage*. Our previous work (Kumar et al., 2018) (referred to as AutoQG hereinafter) generates *candidate answers* from text using Pointer Networks (Vinyals et al., 2015) and *encodes the answer* in the question decoder for improved performance.

We first present a framework in which a *generator* mechanism (the horse) that is employed for generating a question-answer pair invokes or pulls the *evaluator* mechanism (the cart) that is employed for evaluating the generated pair. Our clearly delineated *generator-evaluator* framework lets us (a) easily incorporate several best practices from the above referred previous models in the *generator* while (b) also letting us employ in the *evaluator*, other complex non-decomposable rewards that are consistent with performance measures (such as BLEU and ROUGE) on test data. We also propose some novel reward functions that

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 812–821
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

evaluate the syntax of the question and semantics of the question-answer pair in its entirety. More specifically, since the generated question is in anticipation of some specific answer, we find it most natural to incorporate candidate answer generation (using Pointer Networks) alongside QG right in our generator module, so that the evaluator can optionally take into cognizance the conformity of the generated answer to the ground-truth answer, along with text conformity. Likewise, we also incorporate copy and coverage mechanisms for QG into the generator module so that they can be specifically trained by leveraging a suite of holistically designed and structure-sensitive reward functions in the evaluator module.

The Generator

In Table 1, in rows 1 through 4, we illustrate through examples, the incremental benefits of introducing answer prediction and the copy and coverage mechanisms (See et al., 2017) in the generator. The evaluator associated with the corresponding three generator models employs the conventional and simplistic cross-entropy loss. The motivation for answer prediction in the generator module is obvious and will be further discussed in Section 2.1. In row 3 we illustrate the influence of our copy mechanism, where a rare phrase 'new amsterdam' has been rightly picked up in association with the name of the city.

We however note that in row 3, the word 'new' has been erroneously repeated twice, since an encoder-decoder based model could generate questions with meaningless repetitions.

We introduce a mechanism for discouraging such repetitions in our generator by quantitatively emphasizing the *coverage* of sentence words while decoding. Row 4 shows the improved and relevant question generated by our model trained by incorporating both the copy and coverage mechanisms.

Evaluator

In row 5 of Table 1, we observe the high-quality question that is generated when the simplistic cross-entropy loss in the evaluator is replaced with the more complex and non-decomposable (across words) BLEU reward that accounts for proximity of 'founded' to 'new york'.

In Table 2, we further illustrate the effect of employing other reward functions (described in Section 2.2) in the evaluator. As can be seen, the model that incorporates QG-specific reward functions (QSS and ANSS) generates a significantly better question when compared to the question generated without these rewards.

Limitations of simple decomposable losses: A Seq2Seq model trained using a vanilla cross-entropy loss function (decomposable over words in the question) generates the question *"what year was new york named ?"* (row 1 in Table 1), which is not addressed in the sentence. The passage talks only about the founding of the city and its naming two years later. The inaccuracy of the question is possibly caused by the use of a loss that is agnostic to sequence information. In other words, given its decomposable nature, the cross-entropy loss on the ground-truth question or any of its (syntactically invalid) anagrams will be the same. Moreover, use of the cross-entropy loss in the sequence prediction model could make the process brittle, since the model trained on a specific distribution over words is used on a test dataset with a possibly different distribution to predict the next word given the current predicted word. This creates exposure bias (Ranzato et al., 2015) during training, since the model is only exposed to the data distribution and not the model distribution. Thus, performance suffers due to inadequately evaluating the *structure* of the generated question against the ground-truth question.

The standard metrics for evaluating the performance of question generation models such as BLEU (Papineni et al., 2002), GLEU, and ROUGE-L (Lin, 2004) are based on degree of n-gram overlaps between a generated question and the ground-truth question. It would be desirable to be able to directly optimize these *task-specific metrics*. However, these n-gram based metrics do not decompose over individual words and are therefore hard to optimize. We explicitly employ an evaluator that rewards each generated question based on its conformance to one (or more than one using decomposable attention) questions in the ground-truth set using these possibly non-decomposable reward functions. We find such learning to be a natural instance of reinforcement learning (RL) (Sutton and Barto, 1998) that allows us to use policy gradient to directly optimize task-specific rewards (such as BLEU, GLEU and ROUGE-L), which are otherwise non-differentiable and hard to optimize. In Table 2 we illustrate questions generated using different reward functions. It can be observed that ques-

	Text: "new york city traces its roots to its 1624 founding as a trading post by colonists of the dutch republic and was named new amsterdam in 1626 ."	
Row	**Model**	**Question generated**
1	Seq2Seq model optimized on vanilla (cross entropy) loss without answer prediction	in what 1624 did new york city traces its roots ?
2	Seq2Seq model optimized on vanilla (cross entropy) loss with answer prediction	what year was new york named ?
3	Copy aware Seq2Seq model	what year was new new amsterdam named ?
4	Coverage and copy aware Seq2Seq model	in what year was new amsterdam named ?
5	Seq2Seq model optimized on BLEU (using RL)	what year was new york founded ?

Table 1: Sample text and questions generated using variants of our model.

	Text: "even with the five largest cities in sichuan suffering only minor damage from the quake , some estimates of the economic loss run higher than us $ 75 billion , making the earthquake one of the costliest natural disasters in chinese history ." **Expected answer:** five	
Row	**Model**	**Question generated**
1	GE_{BLEU}	how much did it making for the earthquake of the economic ?
2	$GE_{BLEU+QSS+ANSS}$	how many largest cities in sichuan experience only minor damage from the quake ?
3	GE_{DAS}	how many cities were in sichuan ?
4	$GE_{DAS+QSS+ANSS}$	how many largest cities in sichuan suffering only minor damage from the quake ?
4	GE_{ROUGE}	how much did the economic loss run in sichuan ?
5	$GE_{ROUGE+QSS+ANSS}$	what is the largest cities in sichuan ?

Table 2: Sample text and questions generated using different reward functions, with and without our new QG-specific rewards QSS+ANSS.

tions generated using combination of standard reward functions with reward functions specific to QG quality (QSS+ANSS) exhibit higher quality.

Contributions We summarize our main contributions as follows:

- A comprehensive, end-to-end **generator-evaluator framework** naturally suited for automated question generation. Whereas earlier approaches employ some mechanism (the horse) for generating the question, intertwined with an evaluation mechanism (the cart), we show that these approaches can benefit from a much clearer separation of the generator of the question from its evaluator.

- A *generator* founded on the **semantics** and **structure** of the question by (a) identifying target/pivotal answers (Pointer Network), (b) recognizing contextually important keywords in the answer (copy mechanism), and (c) avoiding redundancy (repeated words) in the question (coverage mechanism).

- An *evaluator* that (a) directly optimizes for conformity to the **structure** of ground-truth sequences (BLEU, GLEU, etc.), and (b) matches against appropriate questions from a set of ground-truth questions (Decomposable Attention).

- Novel reward functions that ensure that the generated question is relevant to the text and conforms to the encoded answer.

When evaluated on the benchmark SQuAD dataset (Rajpurkar et al., 2016), our system considerably outperforms state-of-the-art question generation models (Du et al., 2017; Kumar et al., 2018; Song et al., 2018) in automatic and human evaluation.

2 Our Approach

Our framework for question generation consists of a generator and an evaluator. From the reinforcement learning (RL) point of view, the generator is the *agent* and the generation of the next word is an *action*. The probability of decoding a word $P_\theta(word)$ gives a stochastic *policy*. On every token that is output, an evaluator assigns a reward for the output sequence predicted so far using the current policy of the generator. Based on the reward assigned by the evaluator, the generator updates and improves its current policy. Let us denote the reward (*return*) at time step t by r_t. The cumulative reward, computed at the end of the generated sequence is represented by $R = \sum_{t=0}^{T} r_t$. The goal of our framework is to determine a generator

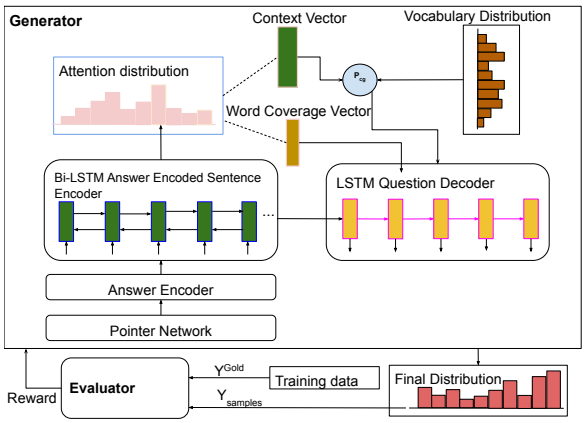


Figure 1: Our generator-evaluator framework for question generation. p_{cg} is the probability which determines whether to copy a word from source text or sample it from vocabulary distribution.

(policy) that maximizes the expected return:

$$Loss_{RL}(\theta) = -\mathrm{E}_{P_\theta(Y_{0:T}|\mathbf{X})} \sum_{t=0}^{T} r_t(Y_t; \mathbf{X}, Y_{0:t-1}) \tag{1}$$

where X is the current input and $Y_{0:t-1}$ is the predicted sequence until time $t-1$. This supervised learning framework allows us to directly optimize task-specific evaluation metrics (r_t) such as BLEU.

The generator is a sequence-to-sequence model, augmented with (i) an encoding for the potentially best pivotal answer, (ii) the copy mechanism (Gu et al., 2016) to help generate contextually important words, and (iii) the coverage mechanism (Tu et al., 2016) to discourage word repetitions. The evaluator provides rewards to fine-tune the generator. The reward function can be chosen to be a combination of one or more metrics. The high-level architecture of our question generation framework is presented in Figure 1.

2.1 Generator

Similar to AutoQG (Kumar et al., 2018), we employ attention and boundary pointer network to identify pivotal answer spans in the input sentence. The generator then takes as input the sequence of words in the sentence, each augmented with encoding of most probable pivotal answer, along with a set of linguistic features such as POS tag, NER tag, *etc.* At each step, the generator outputs a word with the highest probability, to eventually produce a word sequence. Additionally, as we will see, the generator employs copy and coverage mechanisms.

Sentence Encoder: Each word in the input text is fed sequentially into the encoder along with its linguistic features as well as with the encoded pivotal answer (identified by the boundary pointer network). Our encoder is a two-layer bidirectional LSTM network, consisting of $\overrightarrow{h_t} = \overrightarrow{LSTM_2}(x_t, \overrightarrow{h_{t-1}})$ and $\overleftarrow{h_t} = \overleftarrow{LSTM_2}(x_t, \overleftarrow{h_{t-1}})$, which generates a sequence of hidden states. Here x_t is the given input word at time step t, and $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the hidden states at time step t for the forward and backward passes respectively.

Question Decoder: Our question decoder is a single-layer LSTM network, initialized with the state $s = [\overrightarrow{h_t}; \overleftarrow{h_t}]$, which is concatenation of hidden state from forward and backward passes.

We also model the attention (Bahdanau et al., 2014) distribution over words in the source text. We calculate the attention (a_i^t) over the i^{th} source word as $a_i^t = softmax(e_i^t)$, where

$$e_i^t = v^t tanh(W_{eh} h_i + W_{sh} s_t + b_{att}) \tag{2}$$

Here v^t, W_{eh}, W_{sh} and b_{att} are model parameters to be learned, and h_i is the concatenation of forward and backward hidden states of the encoder. We use this attention a_i^t to generate the *context vector* c_t^* as a weighted sum of encoder hidden states: $c_t^* = \sum_i a_i^t h_i$. We further use the c_t^* vector to obtain a probability distribution over the words in the vocabulary as: $P = softmax(W_v[s_t, c_t^*] + b_v)$, where W_v and b_v are model parameters. Thus during decoding, the probability of a word is $P(qword)$. During the training process for each timestamp, the loss is calculated as $L_t = -\log P(qword_t)$. The loss associated with the generated question is:

$$Loss = \frac{1}{T}\sum_{t=0}^{T} L_t = -\frac{1}{T}\sum_{t=0}^{T} \log P(qword_t) \tag{3}$$

2.1.1 The Copy and Coverage Mechanisms:

The copy mechanism facilitates the copying of important entities and words from the source sentence to the question. We calculate $p_{cg} \in [0, 1]$ as the decision of a binary classifier that determines whether to generate (sample) a word from the vocabulary or to copy the word directly from the input text, based on attention distribution a_i^t:

$$p_{cg} = sigmoid(W_{eh}^T c_t^* + W_{sh}^T s_t + W_x x_t + b_{cg}) \tag{4}$$

Here W_{eh}, W_{sh}, W_x and b_{cg} are trainable model parameters. The final probability of decoding a word is specified by the mixture model:

$$p^*(qword) = p_{cg} \sum_{i:w_i=qword} a_i^t + (1-p_{cg})p(qword) \tag{5}$$

Where $p^*(qword)$ is the final distribution over the union of the vocabulary and the input sentence.

As discussed earlier, Equation (5) addresses the rare words issue, since a word not in vocabulary will have probability $p(qword) = 0$. Therefore, in such cases, our model will replace the <unk> token for out-of-vocabulary words with a word in the input sentence having the highest attention obtained using attention distribution a_i^t.

To discourage meaningless multiple repetitions of words in the question (as illustrated in row 3 of Table 1), we maintain a word coverage vector (wcv) for the words already predicted as the sum of all the attention distributions ranging over timesteps 0 until $t-1$. Specifically, at time step t, $wcv = \sum_{t'=0}^{t-1} a^{t'}$.

No word is generated before timestep 0, and hence *wcv* will be a zero vector then. After storing the word coverage vector until $t-1$, while attending to the next word, we will need to inform our attention mechanism about words covered until then. Hence, equation (2) is now modified to be:

$$e_i^t = v^t tanh(W_{wcv} wcv_i^t + W_{eh} h_i + W_{sh} s_t + b_{att}) \tag{6}$$

Here W_{wcv} are trainable parameters that inform the attention mechanism about words that have been previously covered while choosing to attend over the next word. Following the incorporation of the copy and coverage mechanism in our generator, the generator's final loss function will be:

$$Loss_{copy+cov} = -\frac{1}{T}\sum_{t=0}^{T} \log P^*(w_t) + \lambda_c L_{cov} \tag{7}$$

where λ_c is the coverage hyperparameter and the coverage loss L_{cov} is defined as:

$$L_{cov} = \sum_i min(a_i^t, wcv_i^t) \tag{8}$$

We note that this cross-entropy based loss function still does not include task-specific metrics such as BLEU that were motivated earlier. We employ an evaluator to refine the model pre-trained on this loss function to directly optimize the task specific reward. We also empirically show that the refinement of maximum likelihood models using task-specific rewards such as BLEU improves results considerably. In the next subsection we describe our evaluator.

2.2 Evaluator

The evaluator fine-tunes the parameters of the generator network by optimizing task-specific reward functions through policy gradient. It takes as input the predicted sequence and the gold sequence, evaluates a policy, and returns a reward (a score between 0 and 1) that reflects the quality of the question generated. For question generation, the choice of reward functions include task-specific metrics BLEU, GLEU and ROUGE-L (Du et al., 2017; Kumar et al., 2018), as well as the decomposable attention (Parikh et al., 2016) described below. More importantly, we present two new reward functions that are specifically designed for question generation, QSS and ANSS, for the conformity of questions and answers respectively.

Combining Equation (7) with a reward function R (BLEU, GLEU, ROUGE, DAS, QSS and ANSS), we obtain the overall loss function using the expected reward objective as follows:

$$L_{overall} = \alpha * Loss_{copy+cov} - \beta * \sum_{i=0}^{N} \sum_{y \in \mathcal{Y}} P_\theta(y|X^{(i)}) R(y, y^{*(i)}) \tag{9}$$

where $R(y, y^{*(i)})$ denotes per sentence score (reward), $\mathcal{Y}$ is a set of sequences sampled from the final distribution, and α and β are tunable hyperparamters.

2.2.1 Decomposable attention based evaluator

The use of a lexical similarity based reward function such as BLEU or ROUGE does not provide the flexibility to handle multiple possible versions of the ground truth. For example, the questions *"who is the widow of ray croc?"* and *"ray croc was married to whom?"* have almost the same meaning, but due to word order mismatch with the gold question, at most one of them can be rewarded using the BLEU score at the cost of the other(s). Empirically, we find this restriction leading to models that often synthesize questions with

poor quality. We therefore, design a novel reward function, a decomposable attention (Parikh et al., 2016) based similarity scorer (DAS). Denoting by $\hat{q}$ a generated question and by q the ground-truth question, we compute a cross attention based similarity using the following steps:

Cross Attention: The generated question $\hat{q}$ and the ground-truth question q are inter-attended as:

$$\hat{q}_i^* = \sum_{j=0}^{L_q} a_{ji} e(q_j),\ a_{ji} = \frac{\exp(e(\hat{q}_i)^T e(q_j))}{\sum_{k=0}^{L_{\hat{q}}} \exp(e(\hat{q}_i)^T e(q_k))},$$
$$q_j^* = \sum_{i=0}^{L_{\hat{q}}} b_{ji} e(\hat{q}_i),\ b_{ji} = \frac{\exp(e(\hat{q}_i)^T e(q_j))}{\sum_{k=0}^{L_q} \exp(e(\hat{q}_k)^T e(q_j))} \tag{10}$$

where $e(.)$ is the word embedding of dimension size d, $\hat{q}^*$ is the cross attention vector for a generated question $\hat{q}$, and q^* is the cross attention vector for a question q in the ground truth.

Comparison: Each n-gram $\hat{q}_i$ in the generated question (through its embedding $e(\hat{q}_i)$) is compared with its associated cross-attention vector $\hat{q}^*$ using a feed forward neural network N_1. Similarly, each n-gram q_j in the ground-truth question (through its embedding $e(q_j)$) is compared with its associated attention vector q^* using another network N_2 having the same architecture as N_1. The motivation for this comparison is that we would like to determine the soft alignment between n-grams in the generated question and the gold question. As an illustration, while comparing the gold question "*why do rockets look white?*" with a generated question "*why are rockets and boosters painted white?*", we find that an n-gram "*rockets and boosters*" is softly aligned to "*rockets*" while "*look*" is softly aligned to "*painted*".

$$\boldsymbol{\hat{q}_{1,i}} = N_1([e(\hat{q}_i), \hat{q}^*]),\ \boldsymbol{q_{2,j}} = N_2([e(q_j), q^*]) \tag{11}$$

where $\boldsymbol{\hat{q}_{1,i}}$ and $\boldsymbol{q_{2,j}}$ are vectors containing comparison scores of aligned phrases in generated question and gold question respectively and N_1 and N_2 are the feed forward neural nets.

Matching Score: The vectors $\boldsymbol{\hat{q}_{1,i}}$ and $\boldsymbol{q_{2,j}}$ are aggregated over each word or phrase in the predicted question and gold question respectively before feeding them to a linear function (L):

$$DAS = L(\sum_{i=1}^{L_q} \boldsymbol{\hat{q}_{1,i}}, \sum_{j=1}^{L_{\hat{q}}} \boldsymbol{q_{2,j}}) \tag{12}$$

This matching score between the predicted question and the gold question is the reward returned by the decomposable attention based evaluator.

2.2.2 QG quality specific reward functions

We introduce two new reward functions that specifically designed to evaluate the conformity of the generated question (QSS) and answer (ANSS) against the ground truth.

Question sentence overlap score (QSS): This reward function is specific to QG. We compute the sentence overlap score as the number of common n-grams between predicted question and the source sentence. This reward ensures that generated question is relevant to the given sentence. Thus, if $precision_n(s, q)$ computes the $n-$gram precision match between sentence and question,

$$QSS = (\prod_{i=1}^{n} precision_i(sentence, question))^{\frac{1}{n}} \tag{13}$$

Predicted and encoded answer overlap score (ANSS): In order to ensure that the generated question is about the pivotal answer/ground truth answer we calculate answer overlap score. Answer overlap score is the number of common n-grams between the encoded answer and the answer predicted (ans_{qa}) for the generated question using the best performing question answering model over SQuAD[1]

$$ANSS = (\prod_{i=1}^{n} precision_i(\text{ans}_{qa}, pivotal_answer))^{\frac{1}{n}} \tag{14}$$

3 Experimental Setup

In this section, we present our evaluation framework on the publicly available SQuAD (Rajpurkar et al., 2016) dataset. We first explain various reward functions employed in our experiments. We then describe our baseline and the evaluation methods.

Reward Functions: We experimented with the five reward functions discussed in Section 2.2: (1) BLEU, (2) GLEU, (3) ROUGE-L, (4) DAS, and (5) the QG-specific reward QSS+ANSS. In our

[1] https://github.com/huggingface/pytorch-pretrained-BERT

experiments we considered BLEU for up to 4-grams. For the GLEU score, we recorded all subsequences of up to 4-grams.

Baselines and Evaluation Methods: We reimplemented two state-of-the-art question generation models as baselines for comparison: L2A (Du et al., 2017) and AutoQG (Kumar et al., 2018). A direct (and fair) comparison with another recent technique, NQG_{LC} (Song et al., 2018), is not feasible, as unlike us, NQG_{LC} requires ground-truth answers, whereas both AutoQG and our model predict pivotal answers. L2A does not consider answers. Moreover, their context (input is sometimes more than one sentence) is different also the train/test split is different from ours. Hence, we only report the original numbers reported in their paper. We also did not perform human evaluation on NQG_{LC} as their source code has not been made available for reimplementation.

We also use an existing implementation of a recent RL-based abstractive summarization technique (Paulus et al., 2018) to train baseline models SUM_{BLEU} (with BLEU as reward function) and $\text{SUM}_{\text{ROUGE}}$ (with ROUGE as reward function). This comparison studies the effectiveness of state-of-the-art abstractive summarization techniques applied to question generation as-is, as the two are conceptually similar tasks.

We report automatic and human evaluation results on eight variants of our model, each of which is equipped with the copy and coverage mechanism, the pointer network, as well as one of the four reward functions: BLEU, GLEU, ROUGE-L, DAS or one of the four rewards in combination with QG quality specific rewards (QSS+ANSS). Hence, our models are named GE_{BLEU}, etc.

For automatic evaluation, we employ BLEU, ROUGE-L and METEOR, which are standard evaluation measures used to evaluate sequence prediction tasks. We use the evaluation scripts released by (Chen et al., 2015) that was originally used to evaluate the image captioning task.

We also performed human evaluation to further analyze the quality of questions generated for their syntactic correctness, semantic correctness and relevance. Syntactic correctness measures the grammatical correctness of a generated question, semantic correctness measures meaningfulness and naturalness of the question, and relevance measures how relevant the question is to the text. We perform human evaluation for each model on a randomly selected subset of 100 sentences. Each of the three judges is presented the 100 sentence-question pairs for each model and asked for a binary response on each quality parameter. The responses from all the judges for each parameter is then averaged for each model.

3.1 Ablation Analysis

We conducted an ablation analysis to study the effect of removing the copy and coverage mechanisms. Table 4 summarizes the drop in performance for GE_{ROUGE}. Without the copy mechanism, there is a drop overall in every evaluation measure, with BLEU-4 registering the largest drop of 13.8% as against 13.4%, 6.9% and 4.7% in BLEU-3, BLEU-2 and BLEU-1 respectively. On the other hand, without the coverage mechanism, we see a consistent but sufficiently lower drop (1-2%) in each evaluation measure for GE_{ROUGE}.

4 Results and Discussion

We show and compare results on automatic evaluation in Table 3. Note the numbers in parentheses for L2A (Du et al., 2017), AutoQG (Kumar et al., 2018), and NQG_{LC} (Song et al., 2018) are those reported in their original papers. The slight difference of up to 1.7% in the original and reproduced numbers can be attributed to reimplementation and different versions of various libraries used. As can be seen, all our eight models outperform L2A and AutoQG on all evaluation metrics. Two of our models, GE_{GLEU} and GE_{ROUGE}, also outperform NQG_{LC}. Hence, using evaluation metrics as the reward function during reinforcement based learning improves performance for all metrics. We also observe that $\text{GE}_{\text{ROUGE+QSS+ANSS}}$, the model reinforced with ROUGE-L (that measures the longest common sequence between the ground-truth question and the generated question) as the reward function in combination with QG quality specific rewards(QSS+ANSS), is the best performing model on all metrics, outperforming existing baselines considerably. For example, it improves over AutoQG on BLEU-4 by 29.98%, on METEOR by 13.15%, and on ROUGE-L by 8.67%.

In Table 5 we present human evaluation results for the models evaluated on three quality parameters (a) syntactic correctness, (b) semantic correctness, and (c) relevance.

Consistent with automatic evaluation results

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L
L2A (Du et al., 2017)	43.21 (43.09)	24.77 (25.96)	15.93 (17.50)	10.60 (12.28)	16.39 (16.62)	38.98 (39.75)
AutoQG (Kumar et al., 2018)	44.68 (46.32)	26.96 (28.81)	18.18 (19.67)	12.68 (13.85)	17.86 (18.51)	40.59 (41.75)
NQG$_{LC}$ (Song et al., 2018)	-	-	-	- (13.98)	- (18.77)	- (42.72)
SUM$_{\text{BLEU}}$ (Paulus et al., 2018)	11.20-	3.50-	1.21-	0.45-	6.68-	15.25-
SUM$_{\text{ROUGE}}$ (Paulus et al., 2018)	11.94-	3.95-	1.65-	0.082-	6.61-	16.17-
GE$_{\text{BLEU}}$	46.84	29.38	20.33	14.47	19.08	41.07
GE$_{\text{BLEU+QSS+ANSS}}$	46.59	29.68	20.79	15.04	19.32	41.73
GE$_{\text{DAS}}$	44.64	28.25	19.63	14.07	18.12	42.07
GE$_{\text{DAS+QSS+ANSS}}$	46.07	29.78	21.43	16.22	19.44	42.84
GE$_{\text{GLEU}}$	45.20	29.22	20.79	15.26	18.98	43.47
GE$_{\text{GLEU+QSS+ANSS}}$	47.04	30.03	21.15	15.92	19.05	43.55
GE$_{\text{ROUGE}}$	47.01	30.67	21.95	16.17	19.85	43.90
GE$_{\text{ROUGE+QSS+ANSS}}$	**48.13**	**31.15**	**22.01**	**16.48**	**20.21**	**44.11**

Table 3: Experimental results on the test set on automatic evaluation metrics. Best results for each metric (column) are **bolded**. The numbers in parentheses for L2A, AutoQG and NQG$_{LC}$ are those from the best models reported in their respective original papers. The slight difference of up to 1.7% from our reproduced numbers can be attributed to reimplementation and different versions of various libraries used. Models with new QG-specific reward functions (QSS+ANSS) are highlighted in gray for easy comparison.

Model (GE$_{\text{ROUGE}}$)	**Δ BLEU-1 (47.01)**	**Δ BLEU-2 (30.67)**	**Δ BLEU-3 (21.95)**	**Δ BLEU-4 (16.17)**	**Δ METEOR (19.85)**	**Δ ROUGE-L (43.90)**
W/o copy	2.09 (4.7%)	2.13 (6.9%)	2.94 (13.4%)	2.23 (13.8%)	2.21 (11.1%)	2.58 (5.9%)
W/o coverage	0.31 (0.7%)	0.57 (1.9%)	0.94 (4.2%)	0.28 (1.7%)	0.84 (4.2%)	1.01 (2.3%)

Table 4: Ablation analysis results after removing (a) copy mechanism and (b) coverage mechanism from the system (GE$_{\text{ROUGE}}$). Both absolute performance drop and percentage of drop (in parentheses) are reported.

shown in Table 3, seven of our eight models outperform the two baselines, with GE$_{\text{DAS+QSS+ANSS}}$ being the best model on syntactic correctness and semantic correctness quality metrics, outperforming all the other models by a large margin. However, model GE$_{\text{BLEU+QSS+ANSS}}$ generates highly relevant questions and is the best model on relevance metrics.

It is noteworthy that for each of our models (e.g. GE$_{\text{BLEU}}$), adding QG-specific rewards (e.g. GE$_{\text{BLEU+QSS+ANSS}}$) significantly improves question quality in human evaluation, even though there is less noticeable improvements in automatic evaluation. This clearly demonstrates the effectivess of our new QG-specific reward functions.

We measure inter-rater agreement using Randolph's free-marginal multirater kappa (Randolph, 2005). This helps in analyzing level of consistency among observational responses provided by multiple judges. It can be observed that our quality metrics for all our models are rated as *moderate agreement* (Viera et al., 2005).

4.1 Analyzing Choice of Reward Function

BLEU(Papineni et al., 2002) measures precision and ROUGE(Lin, 2004) measures recall, we believe that cross-entropy loss was already accounting for precision to some extent and using it in conjunction with ROUGE (which improves recall) therefore gives best performance.

Model	Syntax		Semantics		Relevance	
	Score	Kappa	Score	Kappa	Score	Kappa
L2A	39.2	0.49	39	0.49	29	0.40
AutoQG	51.5	0.49	48	0.78	48	0.50
GE$_{\text{BLEU}}$	47.5	0.52	49	0.45	41.5	0.44
GE$_{\text{BLEU+QSS+ANSS}}$	82	0.63	75.3	0.68	**78.33**	0.46
GE$_{\text{DAS}}$	68	0.40	63	0.33	41	0.40
GE$_{\text{DAS+QSS+ANSS}}$	**84**	0.57	**81.3**	0.60	74	0.47
GE$_{\text{GLEU}}$	60.5	0.50	62	0.52	44	0.41
GE$_{\text{GLEU+QSS+ANSS}}$	78.3	0.68	74.6	0.71	72	0.40
GE$_{\text{ROUGE}}$	69.5	0.56	68	0.58	53	0.43
GE$_{\text{ROUGE+QSS+ANSS}}$	79.3	0.52	72	0.41	67	0.41

Table 5: Human evaluation results (column "Score") as well as inter-rater agreement (column "Kappa") for each model on the test set. The scores are between 0-100, 0 being the worst and 100 being the best. Best results for each metric (column) are **bolded**. The three evaluation criteria are: (1) syntactically correct (Syntax), (2) semantically correct (Semantics), and (3) relevant to the text (Relevance). Models with new QG-specific reward functions (QSS+ANSS) are highlighted in gray for easy comparison.

DAS calculates semantic similarity between generated question and the gound-truth question. As discussed in section 2.2.1 DAS will give high reward even though the generated question has low BLEU score. Thus, the performance of the

model on automatic evaluation metrics does not improve with DAS as the reward function, though the quality of questions certainly improves. Further, ROUGE in conjunction with the cross entropy loss improves on recall as well as precision whereas every other combination overly focuses only on precision.

Error analysis of our best model reveals that most errors can be attributed to intra-sentence dependencies such as co-references, concept dependencies *etc.* In a camera ready version of the paper, we will share link to a detailed report containing extensive experiments that include ablation tests. Also link to the source code will be provided then.

5 Related Work

Neural network-based methods represent the state-of-the-art in automatic question generation (QG) from text. Motivated by neural machine translation, Du et al (2017) proposed a sequence-to-sequence (Seq2Seq) architecture for QG. In our previous work, we (2018) proposed to augment each word with linguistic features and encode the most relevant *pivotal answer* to the text while generating questions. Similarly, Song et al (2018) encode ground-truth answers (given in the training data), use the copy mechanism and additionally employ context matching to capture interactions between the answer and its context within the passage. They encode ground truth answer for generating questions which might not be available for test set in contrast we train a Pointer Network based model to predict the pivotal answer to generate question about. In our work (Kumar et al., 2019a) we proposed a transformer based architecture to automatically generate complex multi-hop questions from knowledge graphs. In (Kumar et al., 2019b) we proposed a cross lingual training method for automatically generating questions from text in low resource languages.

Very recently deep reinforcement learning has been successfully applied to natural language generation tasks such as abstractive summarization (Paulus et al., 2018; Celikyilmaz et al., 2018) and dialogue generation (Li et al., 2016). In summarization, one generates and paraphrases sentences that capture salient points of the text. On the other hand, generating questions additionally involves determining question type such as what, when, etc., being selective on which keywords to copy from the input into the question, leaving remaining keywords for the answer. This also requires the development of a specific probabilistic generative model. (Yao et al., 2018) proposed generative adversarial network (GAN) framework with modified discriminator to predict question type. Recently Fan et al (2018) proposed a bi-discriminator framework for visual question generation. They formulate the task of visual question generation as a language generation task with some linguistic and content specific attributes.

6 Conclusion

We presented a novel, holistic treatment of question generation (QG) using a generator-evaluator framework. Our generator provisions for explicitly factoring in question syntax and semantics, identifies pivotal answers, recognizes contextually important words and avoids meaningless repetitions. Our evaluator allows us to directly optimize for conformity towards the structure of ground-truth question(s). We propose two novel reward functions account for conformity with respect to ground-truth questions and predicted answers respectively. In conjunction, the evaluator makes use of task-specific scores, including BLEU, GLEU, ROUGE-L, and decomposable attention (DAS) that are naturally suited to QG and other seq2seq problems. Experimental results on automatic evaluation and human evaluation on the standard benchmark dataset show that our framework, especially with the incorporation of the new reward functions, considerably outperforms state-of-the-art systems.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. In *NAACL 2016*, pages 1662–1675. ACL.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading

comprehension. In *ACL*, volume 1, pages 1342–1352.

Zhihao Fan, Zhongyu Wei, Siyuan Wang, Yang Liu, and Xuanjing Huang. 2018. A reinforcement learning framework for natural question generation using bi-discriminators. In *27th International Conference on Computational Linguistics (COLING)*, pages 1763–1774.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*, volume 1, pages 1631–1640.

Vishwajeet Kumar, Kireeti Boorla, Yogesh Meena, Ganesh Ramakrishnan, and Yuan-Fang Li. 2018. Automating reading comprehension by generating question and answer pairs. In *22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*.

Vishwajeet Kumar, Yuncheng Hua, Ganesh Ramakrishnan, Guilin Qi, Lianli Gao, and Yuan-Fang Li. 2019a. Difficulty-controllable multi-hop question generation from knowledge graphs. In *ISWC*.

Vishwajeet Kumar, N. Joshi, Arijit Mukherjee, Ganesh Ramakrishnan, and Preethi Jyothi. 2019b. Cross-lingual training for automatic question generation. In *ACL*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *EMNLP 2016*, pages 1192–1202. ACL.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Prashanth Mannem, Rashmi Prasad, and Aravind Joshi. 2010. Question generation from paragraphs at UPenn: QGSTEC system description. In *Third Workshop on Question Generation (QG 2000)*, pages 84–91.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP 2016*, pages 2249–2255.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP 2016*, pages 2383–2392. ACL.

Justus J Randolph. 2005. Free-marginal multirater kappa (multirater k [free]): An alternative to fleiss' fixed-marginal multirater kappa. *Online submission*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *NAACL (Short Papers)*, volume 2, pages 569–574.

Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *ACL 2016*, pages 76–85. The Association for Computer Linguistics.

Anthony J Viera, Joanne M Garrett, et al. 2005. Understanding interobserver agreement: the kappa statistic. *Fam Med*, 37(5):360–363.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Kaichun Yao, Libo Zhang, Tiejian Luo, Lili Tao, and Yanjun Wu. 2018. Teaching machines to ask questions. In *IJCAI*, pages 4546–4552.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910.

In Conclusion Not Repetition: Comprehensive Abstractive Summarization With Diversified Attention Based On Determinantal Point Processes

Lei Li[1], **Wei Liu**[1], **Marina Litvak**[2], **Natalia Vanetik**[2] and **Zuying Huang**[1]
[1] Beijing University of Posts and Telecommunications
{leili, thinkwee, zoehuang}@bupt.edu.cn
[2] Shamoon College of Engineering
litvak.marina@gmail.com natalyav@sce.ac.il

Abstract

Various Seq2Seq learning models designed for machine translation were applied for abstractive summarization task recently. Despite these models provide high ROUGE scores, they are limited to generate comprehensive summaries with a high level of abstraction due to its degenerated attention distribution. We introduce Diverse Convolutional Seq2Seq Model(DivCNN Seq2Seq) using Determinantal Point Processes methods(Micro DPPs and Macro DPPs) to produce attention distribution considering both quality and diversity. Without breaking the end to end architecture, DivCNN Seq2Seq achieves a higher level of comprehensiveness compared to vanilla models and strong baselines. All the reproducible codes and datasets are available online[1].

1 Introduction

Given an article, abstractive summarization aims at generating one or several short sentences that cover the main idea of original article, which is a combination of Natural Language Understanding(NLU) and Natural Language Generation(NLG).

Abstractive summarization uses Seq2Seq models (Sutskever et al., 2014) which consist of an encoder, a decoder and attention mechanism (Mnih et al., 2014). With attention mechanism the decoder can choose a weighted context representation at each generation step so it can focus on different parts of encoded information. Seq2Seq with attention achieved remarkable results on machine translation (Bahdanau et al., 2014) and other text generation tasks such as abstractive summarizaiton (Rush et al., 2015).

Unlike machine translation that emphasizes attention mechanism as a method of learning word level alignments between source text and target text, attention in summarization should be soft and diverse. Many works noticed that attention may be over concentrated for summarization and hence cause problems like generating duplicate words or duplicate sentences. Researchers try to solve these problems by introducing various attention structures, including local attention (Luong et al., 2015), hierarchical attention (Nallapati et al., 2016), distraction attention (Chen et al., 2016) and coverage mechanism (See et al., 2017) etc. But all these works ignore another repeat problem, as we call it, "Original Text Repetition". We define and explain this problem in section 3.

In this paper we propose a novel Diverse Convolutional Seq2Seq Model(DivCNN Seq2Seq) based on Micro Determinantal Point Processes(Micro DPPs) and Macro Determinantal Point Processes(Macro DPPs). Our contributions are as fol-

Article: marseille , france the french prosecutor leading an investigation into the crash of germanwings flight 9525 insisted wednesday that he was not aware of any video footage from on board the plane . marseille prosecutor brice robin told cnn that so far no videos were used in the crash investigation ..... of a cell phone video showing the harrowing final seconds from on board germanwings flight 9525 as it crashed into the french alps . .....paris match and bild reported that the video was recovered from a phone at the wreckage site . ..... cnn 's frederik pleitgen , pamela boykoff , antonia mortensen , sandrine amiel and anna-maja rappard contributed to this report .
CNN Seq2Seq: french prosecutor UNK robin says he was not aware of any video .
DivCNN Seq2Seq with Micro DPPs: new french prosecutor leading an investigation into the crash of UNK wings flight UNK 25 which crashed into french alps . the video was recovered from a phone at the wreckage site .
DivCNN Seq2Seq with Macro DPPs: french prosecutor says he was not aware of any video footage from on board UNK wings flight UNK 25 as it crashed into french alps .

Table 1: Article-summary sample from CNN-DM dataset. Colored spans are attentive parts. Micro DPPs model puts wider attention on article than vanilla does and Macro DPPs puts the widest attention, including former two models' attentive parts.

[1] available at https://github.com/thinkwee/DPP_CNN_Summarization

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 822–832
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

lows:

- We define and describe the Original Text Repetition problem in abstractive summarization and identify the cause behind it, which are degenerated attention distributions. We have also introduced three article-related metrics for the Original Text Repetition estimation and applied them in our experiments.

- We suggest a solution to this problem in the form of introducing DPPs into deep neural network (DNN) attention adjustment and propose DivCNN Seq2Seq. In order to adapt DPPs to large scale computing, we propose two kinds of methods: Micro DPPs and Macro DPPs. To the best of our knowledge, this is the first attempt to adjust attention distributions considering both quality and diversity.

- We evaluate our models on six open datasets and show its superiority on improving the comprehensiveness of generated summaries without losing much training and inference speed.

2 Convolutional Seq2Seq Learning

Usually encoder and decoder in Seq2Seq architecture are recurrent neural network(RNN) or its variants like Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Chung et al., 2014) network. Recently, a Seq2Seq architecture based entirely on convolutional neural networks (CNN Seq2Seq) (Gehring et al., 2017) was proposed. It has better hierarchical representation of natural language and can be computed in parallel. In this paper we choose CNN Seq2Seq as our baseline system because it performs better on capturing long-term dependency, which is important for summarization.

Both encoder and decoder in CNN Seq2Seq consist of convolutional blocks. Each block contains a one dimensional convolution (Conv1d), a gated linear unit (GLU) (Dauphin et al., 2017) and several fully connected layers for dimension transformation. Residual connection (He et al., 2016) and batch normalization (Ioffe and Szegedy, 2015) are used in each block. Each block receives an input I of size $\mathbb{R}^{B*T*C}$, where B, T, and C are respectively batch size, length of text and number of channels (the same as embedding size). Conv1d pads the sentence first and then generates a tensor $[O_1, O_2]$ of size $\mathbb{R}^{B*T*2C}$, doubling the channel. The extra channels are used in a simple nonlinearity gated mechanism:

$$O_1, O_2 = Conv1d(I) \quad (1)$$

$$GLU([O_1, O_2]) = O_1 \otimes \sigma(O_2) \quad (2)$$

Multi-step attention (Gehring et al., 2017) are used in CNN Seq2Seq. Each convolutional block in decoder has its own attentive context. Followed on query-key-value definition of attention, queries $Q \in \mathbb{R}^{B*T_g*C}$ are different decoder block outputs, where T_g stands for summary length; keys $K \in \mathbb{R}^{B*T_s*C}$ are encoder last block outputs, where T_s stands for article length; values are sum of encoder input embeddings $E \in \mathbb{R}^{B*T_s*C}$ and K. Because of the parallel architecture, attention for all decoder time steps can be calculated at once. Such architecture can speed up training and give convenience for our DPPs calculation. Using the simplest dot product attention, all the calculations can be done with an efficient batch matrix multiplication (BMM).

$$score_{attn} = BMM(Q, K) \quad (3)$$

$$weight_{attn} = Softmax(score_{attn}) \quad (4)$$

$$context = BMM(weight_{attn}, K + E) \quad (5)$$

3 Original Text Repetition

Original Text Repetition(OTR) problem means that each sentence in generated summaries are repetitions of article sentences. The abstractive summarization hence degenerates to extractive summarization. The ROUGE metric can not detect this problem since it only measures the n-grams concurrence between generated summaries and gold summaries without taking articles into consideration. The word repeat problem (See et al., 2017) or the lack of abstraction problem (Kryściński et al., 2018) can be seen as extreme condition or alternative description of OTR. Behind this phenomenon is the degenerated attention distribution learned by model which we define as:

- **Narrow Word Attention** for each summary word, the attention distribution narrows to one word position in article.

- **Adjacent Sentence Attention** for all words in each summary sentence, their positions of

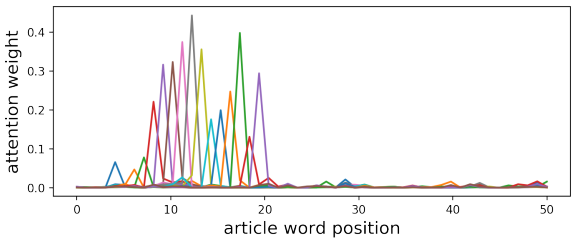


Figure 1: Degenerated attention distribution behind OTR problem. The generated summary repeats the first sentence in article. We select the first 16 words of summary and show their attention over first 50 words of article.

attention peaks are adjacent or semantically adjacent, which means that attended article parts have similar features.

As shown in the Figure 1, sentence attention degenerates to several adjacent peaks on the repeated article positions. Usually each sentence in gold summaries considers multiple article sentences and induct to one, not simply copying one article sentence. The gap between generated summaries(copy) and gold summaries(induce) means that model just learned to find article sentences that has the maximum similarity to gold summaries not the relation between article facts and summaries. Degenerated attention mechanism misleads the model.

4 Diverse Convolutional Seq2Seq Model

To prevent Seq2Seq model from attention degeneration, we introduce DPPs as a method of regularization in CNN Seq2Seq and propose DivCNN Seq2Seq.

4.1 Quality and Diversity Decomposition of Determinantal Point Processes

DPPs have been widely used in recommender systems, information retrieval and extractive summarization systems. It can generate subsets with both high quality and high diversity (Kulesza and Taskar, 2011).

Given a discrete, finite point process P and a ground set D, if for every $A \in D$ and a random subset $\boldsymbol{Y}$ drawn according to P, there is:

$$P(A \in \boldsymbol{Y}) = det(K_A) \tag{6}$$

where K is a real symmetric matrix that indexed by the elements of D, then P is a determinantal point process and K is the marginal kernel of DPPs. Marginal kernel merely gives marginal probability of one certain item to be selected in one particular sampling process, hence we use L-ensemble (Kulesza and Taskar, 2011) to model atomic probabilities for every possible instantiation of Y:

$$K = L(L+I)^{-1} = I - (L+I)^{-1} \tag{7}$$

$$P_L(\boldsymbol{Y} = Y) \propto det(L_Y) \tag{8}$$

$$P_L(\boldsymbol{Y} = Y) = \frac{det(L_Y)}{det(L+I)} \tag{9}$$

L-ensemble is also one kind of DPPs and can be constructed directly using the quality(q) and similarity(sim) of point set:

$$L_{i,j} = q(i) * sim(i,j) * q(j) \tag{10}$$

Equation 9 is a probability that subset Y being chosen, which is actually a quantitative indicator for the score of the subset considering both its quality and diversity(QD-score). Summarization follows the same principle: a good summary should consider both information significance and redundancy. In extractive summarization set of sentences with high score (quality) and diversity is chosen to a summary, using DPPs sampling algorithm (Li et al., 2017).

In Figure 2 we show the difference between quality-only sampling and DPPs sampling. We first generate a simulated attention distribution for testing. Then we use word position distance as similarity measure and attention as quality to construct the L matrix(L-ensemble) for DPPs. Point subset is sampled based on quality (green) or DPPs (blue), then a gaussian mixture distribution was generated around these points to soften and reweight the attention. Both samplings approximate the distribution of original attention distribution (orange), but DPPs approximate it better and have more scattering peaks. Sampling only considering attention weight (quality) generates less peaks, which means many adjacent points with low diversity are sampled.

In actual experiments we choose attention weight as quality. The model learns attention distribution to score different parts of article and obviously higher attention means higher quality. In original CNN Seq2Seq the sum of encoder output and encoder input embeddings are encoded feature vectors. We follow this setting and use the feature vectors to calculate cosine similarity.

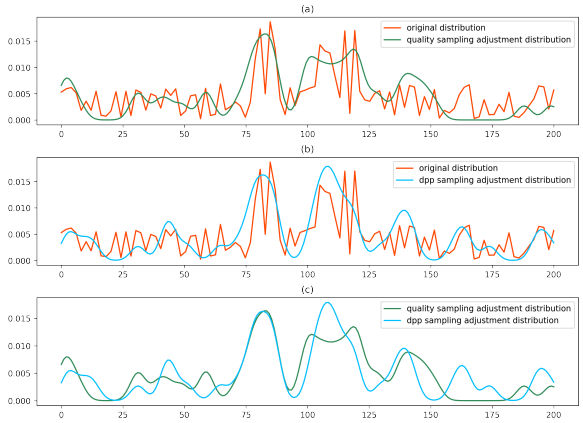


Figure 2: Comparison of different reweighting methods on a simulated distribution. DPPs sampling reweighting approximates original distribution better since it catches the high attention area around position 160. It also samples less adjacent points around position 110.

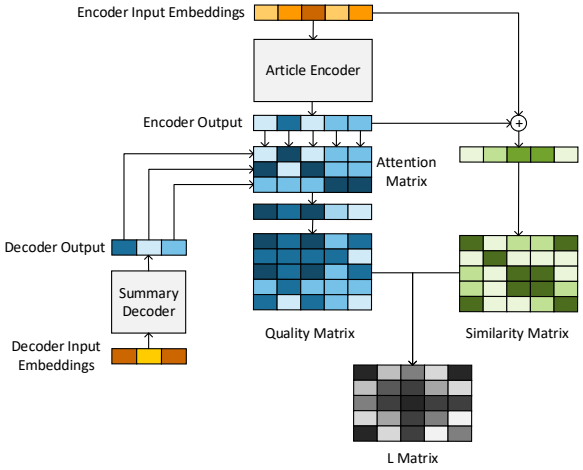


Figure 3: Construction of L matrix

Specifically, the encoder output are tree-like semantic features extracted by CNN encoder while the encoder input embedding provides point information about a specific input element before encoding (Gehring et al., 2017). Hence feature vectors contain both highly abstract semantic features and specific grammatical features when calculating diversity. Compared to extractive summarization, DPPs in abstractive summarization use status of DNN as quality and diversity which can be optimized dynamically during training.

The computation of L matrix is shown in Figure 3. For each sample in a batch(128 in our experiments), the encoder input embeddings $E \in \mathbb{R}^{T_s * C}$ multiply its transpose to produce similarity matrix $S \in \mathbb{R}^{T_s * T_s}$. The weight vectors of Multi-step Attention average over decoder layers and summary length, then do the same operation to generate quality matrix $Q \in R^{T_s * T_s}$. Then we use the hadamard product of Q and S as $L \in R^{T_s * T_s}$.

4.2 Macro DPPs

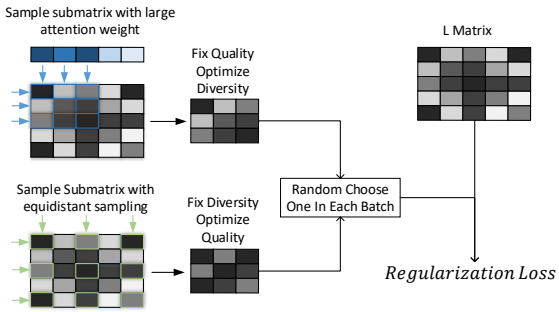


Figure 4: Conditional sampling in Macro DPPs

The idea of Macro DPPs is to pick subsets under some restriction and evaluate QD-score of subset using equation 9. The ideal attention distribution should have subsets with high QD-score.

We do not use DPPs sampling since the purpose of Macro DPPs is to evaluate subsets not to sample subsets with high QD-score. The attention distribute over the ground set so we introduce conditional sampling to sample a subset that has high quality or high diversity, then improve the other metric as follows:

- **Improve Diversity in High Quality Subset** Select points with high attention weight to construct subset and require no gradient for quality matrix, just optimize diversity.

- **Improve Quality in High Diversity Subset** Sampling point subset with high diversity is hard to realize, so we just make equidistant(equidistant on word positions) sampling to approximate it. Contrary to the previous method, we require no gradient for similarity matrix and just optimize quality.

We randomly choose one condition in each batch. After the point subset was chosen, the submatrix L_Y can be built by selecting elements in L indexed by point subset. Then we calculate the QD-score of the submatrix and add it into model loss as a regularization. We calculate the logarithmic summation of eigenvalues to prevent numeri-

cal underflow.

$$loss_{QD} = \sum \log \lambda_{L_Y} - \sum \log \lambda_{L+I} \quad (11)$$

$$\propto \frac{det(L_Y)}{det(L+I)} \quad (12)$$

$$loss_{model} = \gamma loss_{MLE} + (1-\gamma) loss_{QD} \quad (13)$$

4.3 Micro DPPs

The idea of Micro DPPs is to sample a subset Y with large QD-score from all article positions and to use these sampled points as adjusted attention focus points. Then a Gaussian Mixture (GM) distribution around these points is generated as ideal attention distribution($weight_{ideal}$). The whole process can be seen as a selection and softening on attention. The Kullback-Leibler divergence of the ideal distribution and attention distribution ($weight$) then is added into the loss function as regularization.

$$P = BFGMInference(L, t) \quad (14)$$

$$weight_{ideal} = GM_{\mu \in P}(\mu, \sigma, \pi) \quad (15)$$

$$loss_{KL} = KLdiv(weight_{ideal}, weight_{attn}) \quad (16)$$

$$loss_{model} = \gamma loss_{MLE} + (1-\gamma) loss_{KL} \quad (17)$$

Classic sampling algorithm for DPPs (Kulesza and Taskar, 2011) runs slow when the size of L matrix is large and it can not be computed in batch. In the DivCNN Seq2Seq model we need to construct an L matrix for every sample and every layer in the decoder, which is ultimately large. To optimize DPPs runtime for this large-scale computation, we introduce a Batch computation version of Fast Greedy Maximum A Posteriori Inference (Chen et al., 2018)(BFGMInference) to sample a subset with high QD-score.

BFGMInference uses a greedy method to approximate the MAP result $Y_{map} = argmax_{Y \in D}\, det(L_Y)$: each time we select j that has maximum QD-score improvements and add it to Y.

$$f(Y) = \log det(L_Y) \quad (18)$$

$$j = \underset{i \in D \setminus Y}{argmax}\, f(Y \cup \{i\}) - f(Y) \quad (19)$$

Algorithm 1 BFGMInference

Input: matrix $L \in \mathbb{R}^{B*T_s*T_s}$, size of sampled subset t
Output: Sampled subset $Y \in \mathbb{R}^{B*t}$

Initialize $D_i = L_{ii}$; $mask = 1^{B*T_s}$; $J = argmax(\log(D * mask))$; $C \in 0^{B*T_s*1}$
$mask_{j \in J} = 0$
$count = 1$
while $count < t$ **do**
  $candidate = \{i | mask_i = 1\}$
  $ctemp = 0^{B*T_s*1}$, $dtemp = 0^{B*T_s}$
  for $idx = 0; idx < Ts - count; idx++$ **do**
    $i = candidate[:, idx]$, $j = J$
    $e_i = (L_{j,i} - \langle c_j, c_i \rangle)/d_j$
    $ctemp_i = e_i$, $dtemp_i = {e_i}^2$
  end for
  $C = [C, ctemp]$, $D = D - dtemp$
  $J = argmax(\log(D * mask))$
  $mask_{j \in J} = 0$
  $count = count + 1$
end while
$Y = \{i | mask_i = 0\}$
return Y

By using Cholesky decomposition we have:

$$L_Y = VV^T \quad (20)$$

$$L_{Y \cup \{i\}} = \begin{bmatrix} V & 0 \\ c_i & d_i \end{bmatrix} \begin{bmatrix} V & 0 \\ c_i & d_i \end{bmatrix}^T \quad (21)$$

$$Vc_i^T = L_{Y,i} \quad (22)$$

$$d_i^2 = L_{ii} - ||c_i||_2^2 \quad (23)$$

Then we can transform equation 19 into:

$$j = \underset{i \in D \setminus Y}{argmax} \log(d_i^2) \quad (24)$$

The c and d can be updated incrementally according to equation 22 and 23. The complete algorithm is described in Algorithm 1. BFGMInference algorithm gains significant speed improvements when the size of L matrix is large as shown in Figure 5.

5 Experimental Setup

Datasets We test DivCNN Seq2Seq model on the widely used CNN-DM dataset (Hermann et al., 2015) and give detailed analysis on diversity and quality. Also we tried our model on other five abstractive summarization datasets which are NEWSROOM corpus (Grusky et al., 2018), TLDR (Völske et al., 2017), BIGPATENT (Sharma et al., 2019), WIKIHOW (Koupaee and Wang, 2018) and REDDIT (Kim et al., 2018). For CNN-DM corpus we truncate articles to 600 words and summaries to

Dataset	# Docs	Type	Average Document Words	Average Summary Words
CNN-DM	287227/13368/11490	News	789/777/768	55/59/62
NEWSROOM	995041/108862/108837	News	659/654/652	26/26/26
REDDIT	34000/4000/4000	Social Media	418/445/451	20/23/25
BIGPATENT	1207222/67068/67072	Documentation	699/699/699	116/116/116
TLDR	960000/20000/20000	Social Media	197/195/204	19/19/19
WIKIHOW	180000/10000/20000	Knowledge Base	475/488/418	62/60/74

Table 2: Dataset overview(train/valid/test).

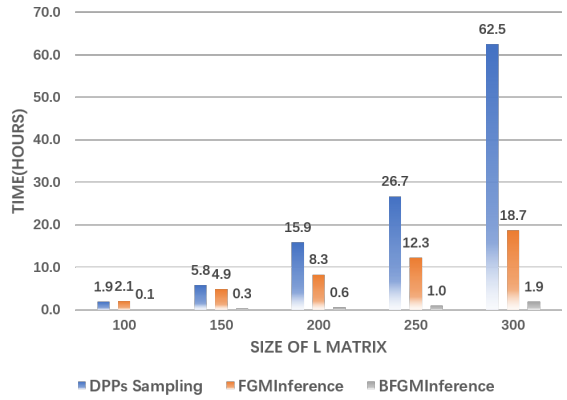


Figure 5: Speed comparison of classical DPPs sampling (blue), FGMInference (red) and BFGMInference (gray) with a batch size of 100.

70 words. For other corpus we only keep articles and summaries that have length around its average length. Specially we only use the TIFU-long version of REDDIT and non-anonymized version of CNN-DM dataset. If the raw datasets were not divided into train/dev/test then we divide the shuffled datasets manually. Details of all six datasets are shown in Table 2.

Hyperparameters and Optimization All the CNN models use a 50000 words article dictionary and 20000 words summary dictionary with byte pair encoding (BPE) (Sennrich et al., 2015). Word embeddings are pretrained on training corpus using Fasttext (Bojanowski et al., 2017; Joulin et al., 2016). We do not train models with large parameters to increase ROUGE results since what we try to improve is the comprehensiveness of each sentence in summary. The total parameters of whole CNN seq2seq Model are about 3800w and the DivCNN Seq2Seq does not change the parameters amount. All the models set embedding dimensionality and CNN channels to 256. The encoder has 20 blocks with kernel size 5 and the decoder has 4 blocks with kernel size 3. Such scale of model parameters are enough for the model to generate fluent summaries. The γ is 0.6 for Macro DPPs and 0.7 for Micro DPPs. In Macro DPPs we choose top 30 points when optimizing diversity and a stride of 20 for equidistant. In Micro DPPs for each summary we sample 20 points to generate gaussian mixture distributions. We train the model with Nesterov's accelerated gradient method using a momentum of 0.99 and renormalized gradients when the norm exceeded 0.1 (Sutskever et al., 2013). The beam search size is 5 and we apply a dropout of 0.1 to the embeddings and linear transform layers. We did not fix training epoches. The model was trained until the average epoch loss can not be lower anymore. The DPPs regularization only works when training and does not bring extra parameters into model. During test the model has learned proper attention so the generation is the same as vanilla CNN Seq2Seq Model.

Article-Related Metrics It is hard to evaluate a summary since summarization itself is very subjective. ROUGE compares generated summaries and gold summaries in checking concurrence of n-grams that results in a very limited evaluation in a word level. We set three article-related metrics to evaluate the comprehensiveness of summaries:

- **Jaccard Similarity Upper Bound (JS)** For each summary sentence, we compute its jaccard similarity with every article sentence. The largest jaccard similarity for each summary sentence is selected as JS. It measures the extent to which summaries copy articles. The worst situation is 1.

- **Sentence Coverage (SC)** We define those article sentences that have a jaccard similarity higher than the gold summaries JS value as covered sentence. Then the average counts of covered article sentences for each summary sentence is a ratio that can be used to measure the coverage of the article by the summaries. The worst situation is less than or equal to 1.

- **Novel Bigram Proportion (NOVEL)** The percentage of bigrams in summaries that did

not appear in the article. It reflects the abstraction of summaries. The worst situation is 0.

Strong Baseline We chose four strong baselines which reported high ROUGE scores. BottomUp (Gehrmann et al., 2018), sumGAN (Liu et al., 2018) and RL Rerank (Chen and Bansal, 2018) are complicate systems that have additional modules or post-processings and partially relieved the OTR problem. The Pointer Generator (See et al., 2017) reaches best ROUGE result in single end to end model but suffers greatly from repetition problem.

Various Possible Causes of the OTR problem We had supposed several other reasons for the repetition problems besides attention degeneration including overfitting, bad usage of translation-style attention mechanism, lack of decoding ability and high variance attention distribution. Respectively, we designed comparative experiments as follows:

- **Overfitting** We set dropout ratio to 0.1 (SMALL) and 0.5 (LARGE) for testing overfitting.
- **Direct Attention** Remove the encoder input embedding in attention value so the decoder looks at highly abstract features directly (DIRECT).
- **Lack of Decoding Ability** We double (DOUBLE) or half (HALF) the vanilla (VANILLA) decoder layers to adjust the decoding ability.
- **High Variance Attention** We scale down the attention distribution manually when training (SCALE), lowering the variance of the distribution.

6 Results

Various Possibilities As shown in Table 4, scaled attention has the lowest jaccard similarity upper bound, which confirms our idea that over-concentrated attention makes model to copy article sentences. As for sentence coverage, small decoder with large drop out ratio performs the best, proving that large and overfitted models may have degenerated attention. Although the scaled attention has the best JS score, its SC score is the worst (SC less than 1.0 means duplicate sentences are generated). So, we may conclude that directly scaling down attention breaks the value of attention. The ideal attention is not about erasing the peak or the variance of attention but to have multiple peaks in sentence attention and have high diversity at the same time. Neither aggregative nor scattering attention distributions do good to summary generation. Direct attention model has the maximum NOVEL score which means point information about a specific input element makes model prefer copying article words instead of generating new words.

CNN-DM Results With large model parameters and dictionaries, four models in strong baselines reach nearly 40 points in ROUGE-1 but they perform poorly on article-related metrics. Single end to end systems like Pointer Generator performs poorly on JS value and NOVEL proportion which means most of its summaries are copied from articles. As for three models with multiple modules or post-processing, the BottomUp model has relatively good jaccard similarity upper bound and the best ROUGE result but its article-related metrics are still far away from gold summaries level. RL Rerank model has better score on JS and sumGAN has better NOVEL score but none of these model reached a balanced good performance on three article-related metrics. Compared to vanilla CNN Seq2Seq, DivCNN Seq2Seq improves the JS and NOVEL points and raises the ROUGE score at the same time, proving that proper attention distribution can help reaching a better local optima. Compared with strong baselines, DivCNN Seq2Seq achieves the best in NOVEL, second and third in JS and SC, respectively. Empirically we suggest that γ for both Micro and Macro DPPs should be set to make average loss change less than 10% compared to vanilla models. We also observed that Micro DPPs is more sensitive to γ compared to Macro DPPs and is easier to converge but may degenerate to vanilla CNN Seq2Seq. Macro DPPs usually can reach better results but it needs more time to train since eigenvalue calculation is expensive and can not be accelerated through fp16 tensor computing.

Novel Bigram NOVEL is a tricky metric which is used in many researches about abstractive summarization. There are many possibilities on explaining a high NOVEL score: first, the summary get a high Novel Bigram ratio because it has many Novel unigrams, which may be good or bad; second, the model may be underfitting and can

Model	JS	SC	NOVEL	R1	R2	RL
gold	0.326	–	0.575	–	–	–
(Liu et al., 2018) sumGAN	0.709	1.136	0.118	39.92	17.65	27.25
(Gehrmann et al., 2018) BottomUp	**0.541**	1.015	0.098	**41.53**	**18.76**	**27.92**
(Chen and Bansal, 2018) RL Rerank	0.585	1.181	0.105	39.38	16.03	24.95
(See et al., 2017) Pointer Generator	0.774	**1.317**	0.079	39.53	17.28	26.89
CNN Seq2Seq Vanilla	0.616	1.137	0.167	30.4	11.7	23.09
DivCNN Seq2Seq with Micro DPPs	0.568	1.214	**0.183**	30.61	11.82	23.19
DivCNN Seq2Seq with Macro DPPs	0.587	1.265	0.177	32.28	12.75	24.32
Extract with Attention	–	–	–	35.48	13.67	22.85
Extract with DPPs Diversified Attention	–	–	–	35.35	13.69	23.07

Table 3: Results on CNN-DM datasets.

Model		JS	SC	NOVEL
SCALE		**0.382**	0.79	0.199
DIRECT		0.567	1.14	**0.207**
LARGE	DOUBLE	0.591	1.201	0.192
	VANILLA	0.639	1.261	0.162
	HALF	0.639	**1.281**	0.153
SMALL	DOUBLE	0.631	1.259	0.167
	VANILLA	0.616	1.137	0.167
	HALF	0.625	1.29	0.161

Table 4: Explore various possible causes of OTR.

not generate fluent sentences; third, the generated summary use novel bigrams to conclude the original text and generate readable sentences, which is the best condition. From the table 3 we can see that Bottom Up has the best ROUGE and JS results but worst NOVEL score. Our DivCNN Models have just the opposite metric scores. These three metrics shouldn't be ambivalent since gold summaries can reach high NOVEL and low JS at the same time. Base on these facts we make the following conjectures:

- Good summaries model learned have styles differ from human-write summaries. Model tend to copy bigrams from original article and reorganize them into short summary sentences. Human tend to use brand new bigrams to paraphrase facts contained in original article. The model use a rewrite(compress and extract) way while human-write is overwrite style.

- Though we use human-write summaries as gold summaries for model to learn and the MLE loss is steadily descending during training but it learned summaries with different style. It implicates that model may not have a "NLU+NLG" process like human do but be restrained in a sentence-level rewrite framework. For Bottom Up and RL Rerank it is not a problem because these two systems are designed to rewrite. They only send parts of article into Seq2Seq. Such design can gain high ROUGE score but it is not the way in which human write gold summaries.

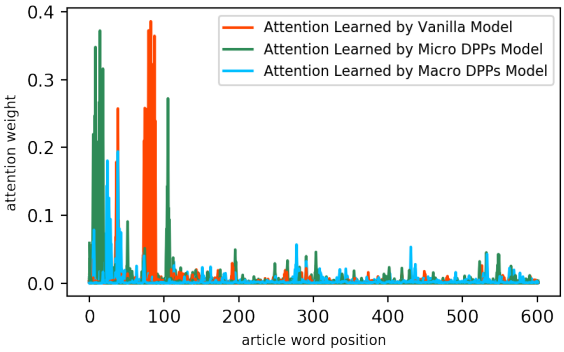


Figure 6: Actual attention distribution learned by vanilla model and DPPs models.

Extractive Summarization based on Learned Attention We also extract article sentences based on sentence attention learned by DPPs models to generate summaries. The attention of a sentence is the sum of the attention weights of the words in the sentence. Table 3 shows that extractive summarization reaches better ROUGE values, implicating that both vanilla and DivCNN models learned appropriate sentence attention. Extractive summarization uses accumulated sentence attention instead of specific distribution, so the results of vanilla models are almost the same as DivCNN.

Sample Visualization We randomly choose one sample in the test set of CNN-DM corpus to visualize and analyze. As shown in Table 1 we

Datasets	Baseline	Baseline	Micro DPPs	Macro DPPs
NEWSROOM	LEAD-3 Baseline	30.63/21.41/28.57	38.33/25.01/35.3	**40.10/26.71/37.24**
TLDR	-	-	65.94/56.97/64.65	**66.93/57.84/65.71**
BIGPATENT	(Chen and Bansal, 2018)	**37.12/11.87/32.45**	33.21/10.47/24.86	34.55/11.65/25.96
WIKIHOW	(See et al., 2017)	**28.53/9.23/26.54**	24.52/6.49/20.56	27.58/8.01/22.61
REDDIT	(Kim et al., 2018)	19/3.7/15.1	21.39/4.24/17.11	**21.57/4.48/17.29**

Table 5: ROUGE F1 results(R1/R2/RL) on different datasets

highlighted attentive parts in the article for different models. The vanilla model just generates one sentence which only focuses on one part of article. The Micro DPPs model generates two sentences considering three parts of the article. Macro DPPs considered article spans that both vanilla model and Micro DPPs model paid attention to. We also checked the attention distribution of this sample. As shown in Figure 6, vanilla model (red) learned only several peaks over article position 70 to 90, which suggests that it only focuses on one sentence and repeats this sentence in a summary. Attention learned by Micro DPPs model (green) still narrows to several peaks but explores more positions compared to vanilla. Macro DPPs (blue) has more natural design of loss function and it optimize quality and diversity directly so it has a more scattering attention distribution.

More Datasets We test our model on other five newly-released abstractive summarization datasets which have various compression ratio, different professional field and more flexible human-write summaries. Only ROUGE results are collected since no baseline generated summaries are provided for us to calculate article-related metrics. Table 5 shows that DivCNN performs better than best baselines on NEWSROOM, REDDIT and reaches incredible ROUGE scores more than 60 (but no baseline is reported in the dataset paper so the result is not comparable). The compression ratio and article length have little impact on the performance of DivCNN. The results show that DivCNN prefers short summaries.

Attention & Representation Degeneration In order to solve attention degeneration we introduce DPPs to improve the diversity of features where model paid high attention to. This solution is consistent with the Presentation Degeneration Problem in NLG (Gao et al., 2018). As shown in Figure 7, Macro DPPs have more diverse embedding presentation compared to vanilla model. (Gao et al., 2018) directly add a regularization loss of diversity to increase the representation power of word embeddings while we aim at generating attention distribution considering both quality and diversity, resulting in learning word embeddings with rich representation power.

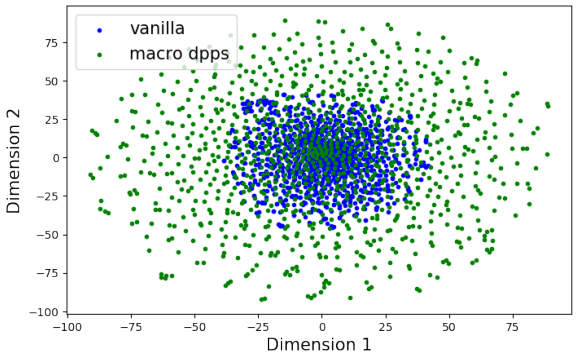


Figure 7: Presentation Degeneration Problem in NLG. We use tSNE (Maaten and Hinton, 2008) to reduce the dimension of word embeddings learned in the model.

7 Conclusions and Future Works

We have defined the "OTR" problem that leads to incomplete summaries and revealed the cause behind it, which is attention degeneration. We also introduce three article-related metrics to evaluate this problem. DPPs are applied directly on attention generation and we propose Macro and Micro DPPs versions of DivCNN Seq2Seq model to adjust attention considering both quality and diversity. Results on CNN-DM and other five open datasets show that DivCNN Seq2Seq can improve the comprehensiveness of summaries.

Due to the hardware limitation we only train a small-parameters version of DivCNN. Also we lost some precision when approximating L matrix and accelerating sampling. These drawbacks lead to limited performance improvements. In the future we hope to explore further on following directions: Quantifiable and controllable quality/diversity in DPPs; better approximation in conditional sampling, such as dynamic sampling stride adjustment; try to apply DPPs-optimized attention on another student model to improve generation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Laming Chen, Guoxin Zhang, and Eric Zhou. 2018. Fast greedy map inference for determinantal point process to improve recommendation diversity. In *Advances in Neural Information Processing Systems*, pages 5622–5633.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for document summarization. *arXiv preprint arXiv:1610.08462*.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *arXiv preprint arXiv:1805.11080*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 933–941. JMLR. org.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tieyan Liu. 2018. Representation degeneration problem in training natural language generation models.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. *arXiv preprint arXiv:1808.10792*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *arXiv preprint arXiv:1804.11283*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2018. Abstractive summarization of reddit posts with multi-level memory networks. *arXiv preprint arXiv:1811.00783*.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*.

Wojciech Kryściński, Romain Paulus, Caiming Xiong, and Richard Socher. 2018. Improving abstraction in text summarization. *arXiv preprint arXiv:1808.07913*.

Alex Kulesza and Ben Taskar. 2011. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1193–1200.

Lei Li, Yazhao Zhang, Junqi Chi, and Zuying Huang. 2017. Uids: A multilingual document summarization framework based on summary diversity and hierarchical topics. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 343–354. Springer.

Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2018. Generative adversarial network for abstractive text summarization. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Eva Sharma, Chen Li, and Lu Wang. 2019. Bigpatent: A large-scale dataset for abstractive and coherent summarization. *arXiv preprint arXiv:1906.03741*.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63.

Generating Formality-tuned Summaries Using Input-dependent Rewards

Kushal Chawla*
University of Southern California
Los Angeles, CA, USA
kchawla@usc.edu

Balaji Vasan Srinivasan, Niyati Chhaya
Adobe Research
Bangalore, India
balsrini,nchhaya@adobe.com

Abstract

Abstractive text summarization aims at generating human-like summaries by understanding and paraphrasing the given input content. Recent efforts based on sequence-to-sequence networks only allow the generation of a single summary. However, it is often desirable to accommodate the psycho-linguistic preferences of the intended audience while generating the summaries. In this work, we present a reinforcement learning based approach to generate formality-tailored summaries for an input article. Our novel input-dependent reward function aids in training the model with stylistic feedback on sampled and ground-truth summaries together. Once trained, the same model can generate formal and informal summary variants. Our automated and qualitative evaluations show the viability of the proposed framework.

1 Introduction

Efficient content consumption is not only driven by the contained information but also by the tone/style of the presentation. The style in text is its non-informational or non-factual aspect and usually drives the quality of response from its audience. A *persuasive* snippet or a teaser might drive up sales for a marketing message and a piece of *formal* text will appeal better to corporate executives as against informal communication. Similarly, not all long form content is easy to read. A succinct representation of the content, i.e. the summary, plays an important role for its quick and efficient consumption. While, text summarization and to some extent *style-understanding* have been independently studied, approaches to generate style-tailored summaries are limited.

Models for style predictions (Pavlick and Tetreault, 2016; Brooke et al., 2010; Danescu-Niculescu-Mizil et al., 2013) are limited to measuring style in text. There have been multiple attempts towards transfer of style (Artetxe et al., 2017; Han et al., 2017; Shen et al., 2017; Tikhonov and Yamshchikov, 2018). Rao and Tetreault (2018) introduced a parallel corpus for formality style transfer with neural machine translation benchmarks. Niu et al. (2017) generate automatic translations tailored towards formality. However, none of these approaches account for the length/succinctness of the created content, and hence do not address the stylized summarization task. In this work, we propose an approach to generate summaries while simultaneously tailoring towards formality preferences (Table 1).

Tunable or controlled summary generation has picked up pace in recent times. Algorithms allow for controlling various dimensions of the output summary such as the length or entities (Fan et al., 2017) and topics (Krishna and Srinivasan, 2018). Since these approaches primarily rely on the diversity in the given dataset, extending these approaches for formality tailored summarization would require a diverse summarization corpus that captures subtleties in various formal variants. Since such a dataset is difficult to curate, it is non-trivial to use these methods as is. Reinforcement learning (RL) based loss functions have recently shown promise in tuning the output on rewards such as ROUGE (Paulus et al., 2018). In such methods, the model receives explicit feedback on the sampled sequences while training. If directly applied for controlling stylistic parameters like formality, such a method would need two separately trained models for generating formal and informal summaries and thus, may miss out on the common learnings.

In this work, we propose a method to incor-

* Work done when author was a full time employee at Adobe Research

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 833–842
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

porate formality in abstractive text summarization. To build a formality-rich training dataset, we merge data from two domains: news and social media - the first one representing more formal language and the latter, informal. We define a novel input-dependent reward function which aids in training the model with stylistic feedback on sampled and ground-truth summaries together. Once trained, the same model can generate formal and informal summary variants. We show the effectiveness of our approach through automated and crowd-sourced experiments, evaluating both the quality and formality levels of the generated summaries. Table 1 shows sample formal and informal summary variants, generated from our approach on an instance from the testset.

Input Article: katrina had been expressing anxiety for a while now about how worried she was about the bridal shower and being the center of attention of a bunch of people she did n't know . that 's completely normal . she has been so worried that she determined to send him a short email outlining what she would do if she were in his shoes . that absolutely counts as meddling . that 's literally the definition of meddling sticking your nose where it does n't belong . katrina talked to my mom for about five minutes and then sat about twenty feet away from her during this tournament while my mom sat alone . and katrina is n't obligated to entertain your mother . your mom could 've talked to other people instead ...
Informal: katrina had been so worried that she was abt the bridal shower & being the center of attention of a bunch of people she did n't know . katrina 's n't obligated to entertain ur mom .
Formal: katrina had been expressing anxiety for the bridal shower and being the center of attention of a bunch of people she did not know . she has been therefore worried that she determined to transmit him a short email outlining what she were in his shoes .

Table 1: Example formal and informal summary variants generated on an instance from our testset.

2 Related Work

Early abstractive summarization efforts were either template-based (Wang and Cardie, 2013; Genest and Lapalme, 2011) or employed ILP-based sentence compression (Filippova, 2010; Berg-Kirkpatrick et al., 2011; Banerjee et al., 2015). With the advent of deep sequence-to-sequence models (Sutskever et al., 2014), attention-based neural models have been proposed for long text summarization (Rush et al., 2015; Chopra et al., 2016). Recent approaches (Nallapati et al., 2017; See et al., 2017) have focused on larger datasets such as the CNN/DailyMail corpus (Hermann et al., 2015; Nallapati et al., 2016). Gulcehre et al. (2016) introduced the ability to copy out-of-vocabulary words from the article to incorporate rarely seen words like names in the generated text. Tu et al. (2016) included the concept of coverage, to prevent the models from repeating the same phrases while generating a sentence. See et al. (2017) proposed a pointer-generator framework which incorporates these improvements, and also learns to switch between generating new words and copying them from the source article. We use this pointer-generator framework as the underlying architecture.

2.1 Incorporating additional constraints

Controlled summary generation has only recently gained popularity. Variational auto-encoders (Hu et al., 2017) or adversarial training (Shen et al., 2017) have been explored for non-parallel stylistic text generation. Sennrich et al. (2016) propose modifications to neural machine translation to tune the level of politeness in the generated text. Ficler and Goldberg (2017) use a conditional language model to control variations like descriptiveness and sentiment simultaneously during generation. Efforts for constrained text summarization are rather limited with no efforts attempting to incorporate psycho-linguistic preferences. Krishna and Srinivasan (2018) incorporate input topic information in the output summary using a topic-vector along with the input word sequence.

Fan et al. (2017) control length and entity in textual summaries using explicit input indicators or tokens. Paulus et al. (2018) directly control the ROUGE evaluation metric using the Self Critical Sequence Training (SCST) (Rennie et al., 2017) algorithm. We build on these approaches and show through our experiments that our approach is able to generate better formality-tuned summaries in comparison to these methods.

2.2 Incorporating Formality

Formality is an important style or tone dimension in written text. Although there are existing works which model formality in text (Brooke et al., 2010; Lahiri, 2015; Pavlick and Tetreault, 2016; Chhaya et al., 2018), there have been limited attempts to incorporate it in text generation. Sheikha and Inkpen (2011) used a predefined set of rules based on formal-informal parallel lists to generate formal and informal sentences. More recently, a parallel corpus of formality style transfer (Rao and Tetreault, 2018) was released with NMT-based benchmarks. To the best of our knowledge, we are the first to introduce formality in the space of abstractive text summarization.

Generating text with varying levels of formality was studied recently in Machine Translation (Niu et al., 2017, 2018). A re-ranking mechanism on

the decoded hypotheses is used to control the formality of the generated translations. We augment our decoding module with a similar, but simpler re-ranking method to enhance our approach.

3 Pointer-Generator Framework

Our approach uses the pointer generator network (See et al., 2017) as the underlying architecture. This section explains this framework briefly for the sake of completion. The model is based on an encoder-decoder setup. The bi-directional LSTM encoder takes the article x as an input and computes a sequence of encoder hidden states $h_1, h_2, ..h_n$. The last state h_n becomes the initial state of the LSTM decoder which uses an attention mechanism to generate the output summary word by word. Further, at each time step, the decoder network computes p_{gen}, the probability of generating a new word from the vocabulary,

$$p_{gen} = \sigma(w_h^T h_t^* + w_s^T s_t + w_x^T x_t + b_{gen}) \quad (1)$$

where w_h, w_s, w_x, b_{gen} are trainable parameters. h_t^* is the context vector capturing the attention distribution, s_t is the decoder internal state and x_t is the decoder input at t^{th} time step. The total probability of w being the next word generated in the summary, $p(w)$, is given by,

$$p(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^t \quad (2)$$

where p_{gen} provides a switch between generating a new word from $P_{vocab}(w)$ or copying a word from the input based on the attention distribution. The training loss is set to be the average negative log-likelihood of the ground truth summary:

$$L_{nll} = -\frac{1}{T} \sum_{t=1}^{T} \log \left[p(y_t^* | y_1^*, ..., y_{t-1}^*, x) \right] \quad (3)$$

where y^* is the ground-truth word sequence. We propose to modify the above training objective in order to incorporate psycho-linguistic preferences of the target audience, with a focus on formality.

4 Generating (In)formal Summaries

A major challenge in incorporating formality into the summarization system is the lack of a formality-diverse dataset. To the best of our knowledge, there exists no data which either has both formal and informal ground-truth summaries for the same input article or where the provided ground-truth summaries are diverse on the formality scale. This makes the direct use of explicit indicators ineffective (Section 6.2), which have been shown to capture this diversity in the given dataset well (Fan et al., 2017; Krishna and Srinivasan, 2018). To address this, we work off a dataset mixed from two different domains: news and social, making it more formality-diverse (Section 5).

While diversity in the data helps the model to learn the (in)formal parts in the text, we further employ a modified reinforcement learning approach which teaches the decoder module to write (in)formally through explicit feedback. The model is trained using feedback on the formality of both sampled and ground-truth summaries together. The pointer-generator model is trained using the negative log-likelihood loss L_{nll} as given in Equation 3. We make the use of policy gradients by introducing an additional loss term L_{rl} in the training objective:

$$L_{rl} = -\left[r(y^s) \right] \sum_{t=1}^{T} \log \left[p(y_t^s | y_1^s, .., y_{t-1}^s, x) \right], \quad (4)$$

where r is the formality-based reward function. y^s is a sampled word sequence generated by sampling from the $p(y_t^s | y_1^s, y_2^s, .., y_{t-1}^s, x)$ distribution at each time step. In essence, optimizing L_{rl} improves the expected reward of the generated output. The final loss is a linear combination of negative log-likelihood loss with the RL loss,

$$L = (1 - \alpha) \cdot L_{nll} + \alpha \cdot L_{rl}, \quad (5)$$

where α governs the strength of the RL-based loss term.As we will show in Section 4.2, we define L_{rl} based on the input tokens and thus the same trained model can generate formal and informal summary variants for a given input article.

While decoding, our framework employs a beam search algorithm to explore plausible outputs (or hypotheses) for generating the final summary. It finally outputs the hypothesis with the maximum probability of generation. However, we observe a reasonable difference in the formality scores among hypotheses with similar generation probabilities. As a part of post-processing, we therefore employ hypotheses re-ranking to further strengthen our generation following Niu et al. (2017). We pick the hypothesis with the maximum formality score among the k hypotheses with

highest generation probabilities. We also perform word-replacement using parallel formal-informal word lists curated from (Sheikha and Inkpen, 2011). This helps to tackle unwanted formal or informal words which must have been copied directly from the input article by the pointer generator frameworks (See et al., 2017). As we show later, this helps our model to better capture formality oracle.

4.1 Measuring Formality in Text

The first step towards defining our reward function is to construct an oracle to measure formality. We define formality using lexical scores (Brooke et al., 2010; Brooke and Hirst, 2014) leveraging the implementation by Niu et al. (2017) who incorporate formality into Machine Translation[1]. They report best performance using a combination of a Support Vector Machine (SVM) model with Word2Vec representations on the CTRW (Hayakawa and Ehrlich, 1994) and BEAN (Lahiri, 2015; Pavlick and Tetreault, 2016) datasets, obtaining 84.4% accuracy on the former and a Spearman's ρ of 0.662 on the latter.

First, two sets of seed words are chosen, which represent formal and informal language respectively. We use the lists curated by Sheikha and Inkpen (2011). They contain various abbreviations with their full forms and 'informal:formal' semantically similar word pairs such as 'about:approximately', 'copy:replica', 'risk:jeopardy', and 'tasty:palatable'. We combine these word lists to create a total of 667 formal and informal seeds. Next, an SVM model is trained to find a separating hyperplane between vector space representations (coming from Word2Vec model trained on Google News corpus) of these formal and informal seeds. Once the model is trained, for any given word, Euclidean distance to this hyperplane is used as a measure of word level formality. To compute the formality of a word sequence y, we use the weighted average function from Niu et al. (2017):

$$F(y) = \frac{\sum_{w_i \in y} |L(w_i)|.L(w_i)}{\sum_{w_i \in y} |L(w_i)|}, \qquad (6)$$

where $L(w_i)$ is the lexical formality score from the SVM model described above.

$F(y)$ represents the formality of the word sequence y, where larger positive values correspond to higher levels of formality and negative values represent informality in text. Using this measure for formality as an oracle, our objective now is to teach the model the intricacies of high and low levels of formality and ultimately, taking this into consideration while summary generation. This is achieved through the reward function r, which is described next.

4.2 Defining the reward function r

We propose an indicator-based rewarding setup to simultaneously benefit from the common learnings of the two models (formal and informal summaries) and incorporate feedback on the ground-truth summaries in the dataset, as against using the formality oracle $F(y)$ described in Section 4.1 directly as separate reward function for formal and informal models.

We use explicit indicators along with the input sequence and set the reward function accordingly. For training, first the oracle $F(y)$ from equation 6 is used to classify ground-truth summaries in the dataset into informal, neutral, or formal classes. We then assign two vocabulary ids (called tokens or indicators) to each class. While training, these tokens are added to the beginning and end of the input article, based on the formality class of the corresponding ground-truth summary. For instance, for a given (article, summary) pair (a, s) in the training dataset, if s is classified as formal, we add the corresponding two tokens at the beginning and end of the input a. The usage of tokens in this manner acts as indicators, providing the model with feedback on the ground-truth summary in the training stage. The usage of two tokens keeps the input **symmetric**, making it easier for both the forward and backward LSTM encoder networks to absorb the formality information at the start of generating their half of the encoding states. We then determine our reward function for RL loss L_{rl} based on the formality class of the ground-truth summary:

$$r(y^s) = \begin{cases} F(y^s) & \text{for formal } y^* \\ 0.0 & \text{for neutral } y^* \\ -F(y^s) & \text{for informal } y^* \end{cases} \qquad (7)$$

where y^* is the ground-truth summary sequence and $F(.)$ is the score from Equation 6. If the ground-truth summary is formal (as denoted by the corresponding tokens), our reward works to maximize the expected formality of the output sum-

[1] https://github.com/xingniu/computational-stylistic-variations

mary and minimize it in case of informal ground-truth summaries. Given an unseen input article, the two tokens can be added to the input sequence based on the type of output summary required. An input-dependent reward function allows the same model to generate both the summary variants, unlike the vanilla framework, which loses the opportunity to learn the commonalities in the two spaces.

4.3 Similarity to SCST method

Equation 4 employs REINFORCE algorithm (Williams, 1992) and can be seen as a modification to the Self Critical Sequence Training (SCST) (Rennie et al., 2017) which was applied to successfully optimize on ROUGE in summarization (Paulus et al., 2018). In SCST, the word with the maximum probability is greedily chosen from the output distribution at each time step, forming a greedy sequence y^b. The model uses the reward of y^b as a baseline in the loss function. Instead, our formulation can be seen as using the baseline reward of 0. This compares the formality level of the sampled sequence y^s with the perfectly neutral summary (with formality score 0.0), penalizing any sequences lying on the opposite side of the desired formality levels.

5 Experimental Setup

We evaluate our approach on its ability to generate formal and informal summaries for an input article. We compare it with several baselines using both automated metrics and crowd-sourcing experiments.

Dataset: A combined dataset from 2 domains: news and social media is used. For the former, we use the CNN/DailyMail news dataset (Hermann et al., 2015; Nallapati et al., 2016), widely used for the task of abstractive text summarization. For the latter, we use the Webis-TLDR-17 corpus (Völske et al., 2017), automatically created using $TL;DR$ tags on Reddit[2]. Figure 1 shows the distribution of lexical formality scores over these and the complete dataset (based on Equation 6). As depicted, the combination allows us to ensure that the dataset contains formality-diverse 'article:summary' pairs. For CNN/DM, the average formality is -0.097, with minimum as -2.451 and maximum as 2.62. For Reddit dataset, the average formality is -1.068, minimum -2.653 and maximum is 2.783. While the average values are negative, through a manual analysis, we found the summaries with more than -0.5 to be reasonably formal in general. We refer the readers to Section A in Supplementary material where we show some sample ground-truth summaries in the dataset along with their formality scores.

The news dataset contains $287,226$ training instances, $13,368$ validation and $11,490$ test instances. The articles have an average length of 781 tokens and multi-sentence summaries with average length of 56 tokens. We use these average values to extract a similar-sized subset of 4 million data points in the Reddit dataset and merge them with the news dataset. We filtered out poor summaries in Reddit dataset heuristically. Several summaries which contain edit: actually refer to additional information not in the article. We filter out summaries containing such keywords. Keeping only the most formal and informal pairs, the training dataset reduces to $286,358$ input-output pairs. $10,000$ pairs are held out for validation and testing, each containing data points from both domains.

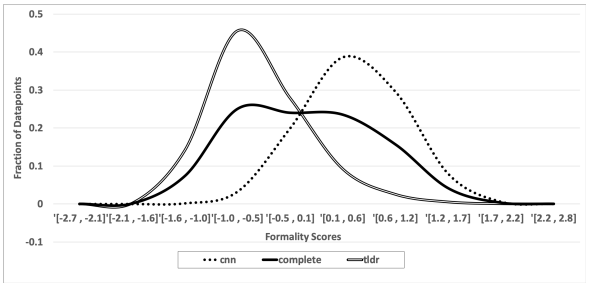


Figure 1: Distribution of lexical formality (Equation 6) in CNN/Daily Mail, Reddit and the complete dataset. Positive values on the X-axis indicate high formality and negative values indicate informality.

Hyperparameters: All methods are implemented using the pointer-generator framework described in Section 3. Following See et al. (2017), the network uses 256 hidden dimensions, embedding size as 128, vocabulary size as $50,000$, 400 maximum encoding steps and 100 maximum decoding steps. We use these hyper-parameters for all the approaches. All our models train for approximately $50,000$ iterations using a batch of size 16.

6 Automated Evaluation

We report the F1 scores for ROUGE-1, ROUGE-2, and ROUGE-L metrics, evaluating how close the generated summaries are to the reference sum-

[2]https://www.reddit.com/

	Informal Summaries					Formal Summaries				
Method	**ROUGE F-score**			**Formality**	**Improvement**	**ROUGE F-score**			**Formality**	**Improvement**
	1	**2**	**L**		(%) vs **PGen**	**1**	**2**	**L**		(%) vs **PGen**
SymVoTo (A)	**26.95**	**8.23**	**23.28**	-0.560 ± 0.902	$+6.66$	**27.00**	**8.15**	**23.23**	-0.432 ± 0.822	$+17.71$
ZeroRL (B)	24.03	6.90	21.08	-0.581 ± 0.925	$+10.66$	24.42	6.93	20.90	-0.358 ± 0.760	$+31.80$
A+B	24.75	7.27	21.49	-0.603 ± 0.923	$+14.85$	25.72	7.51	22.20	-0.399 ± 0.801	$+24.00$
Our Approach	23.02	6.54	20.19	$\mathbf{-0.727 \pm 0.846}$	$\mathbf{+38.47}$	24.8	6.72	21.58	$\mathbf{-0.052 \pm 0.738}$	$\mathbf{+90.09}$

Table 2: Performance of various ablations of the proposed approach on automated evaluation metrics in generating informal and formal summaries. Formality score is computed from the oracle (Eq. 6), averaged over the testset. **SymVoTo** refers to the use of two vocabulary tokens in a symmetrical manner, **ZeroRL** refers to the use of Zero baseline reward instead of Greedy baseline (Section 4). **A+B** combines these two methods and the complete approach further employs post-processing steps.

	Informal Summaries					Formal Summaries				
Method	**ROUGE F-score**			**Formality**	**Improvement**	**ROUGE F-score**			**Formality**	**Improvement**
	1	**2**	**L**		(%) vs **PGen**	**1**	**2**	**L**		(%) vs **PGen**
PGen	**26.96**	**8.18**	**23.18**	-0.525 ± 0.875	-	**26.96**	**8.18**	**23.18**	-0.525 ± 0.875	-
VoTo	26.61	8.17	22.99	-0.556 ± 0.883	$+5.90$	26.76	8.13	23.04	-0.452 ± 0.821	$+13.90$
StyleSum	14.84	2.06	13.03	$\mathbf{-0.762 \pm 0.815}$	$\mathbf{+45.14}$	11.18	0.63	9.93	-0.59 ± 0.800	-12.38
GreedyRL	26.00	7.53	22.39	-0.476 ± 0.827	-9.33	26.47	7.92	22.72	-0.458 ± 0.829	$+12.76$
Our Approach	23.02	6.54	20.19	-0.727 ± 0.846	$+38.47$	24.8	6.72	21.58	$\mathbf{-0.052 \pm 0.738}$	$\mathbf{+90.09}$

Table 3: Performance based on Automated Evaluation Metrics for Generating Informal and Formal Summaries. Formality score is computed from the oracle (Eq. 6), averaged over the testset. All the prior approaches were adapted for formality, as described in Section 6.2.

maries. To evaluate the efficacy of the methods in capturing formality, we report the average formality in the output summaries and corresponding percentage improvements in average formality, relative to **PGen** (See et al., 2017).

6.1 Ablation study

We performed an ablation study over our approach to analyze the effect in performance by the use of two symmetric vocabulary tokens (**SymVoTo**) and a zero-reward baseline (**ZeroRL**) separately.

For training in **SymVoTo**, three levels of formality are defined based on the scores from the formality oracle $F(y)$ (Equation 6): informal (less than -0.2), neutral (between -0.2 and $+0.2$), and formal (greater than 0.2). In our training dataset, $169,628$ summaries were tagged as informal, $31,129$ as neutral and $85,601$ as formal. While training, the two tokens are added to the input article based on the formality level of the ground-truth summary. To generate the formal or informal summaries for an unseen article, we add the corresponding two tokens to the input and pass it through the trained model.

The **ZeroRL** method is trained using the joint objective in Equation 5. However, instead of using the input-dependent reward function in Equation 7, it directly optimizes on the formality oracle. Due to the slow training speeds with policy learning, we first pre-train our network with pointer-generator method (Section 3). We further train the model with policy learning for 3000 iterations. We use a fixed weight of 0.9 for L_{rl} in Equation 5 and 0.1 for negative log likelihood loss L_{nll}. To generate formal and informal summaries in this case, we train two separate models, one where the reward function is $F(y)$ to maximize the expected formality, and second, in which the reward function is $-F(y)$, to minimize the expected formality.

Training for our own approach is similar to the vanilla RL method described above, but with three differences. First, in order to use input-dependent rewards (Equation 7), we first pre-train the model with **SymVoTo** model instead of the pointer-generator method. Secondly, once pre-trained, we optimize on our input-dependent reward function instead of directly using the formality oracle. Finally, once completely trained, the same model can be used to generate formal and informal summaries by supplying the corresponding tokens at the input. While decoding these respective summaries, we use k=4 for hypothesis reranking.

The results of this study are shown in Table 2. The models **SymVoTo** and **ZeroRL** are independently able to beat the baseline from Table 3 in capturing formality. Our approach which combines these two methods using input-dependent rewards and further employs post-processing is able to better capture the formality oracle.

6.2 Evaluation against existing baselines

Multiple style transfer and summarization models are adapted for this task as baselines. Our first baseline is the vanilla, pointer-generator network described in Section 3. It generates a single, generic summary, without using any formality information. We refer to it as **PGen**.

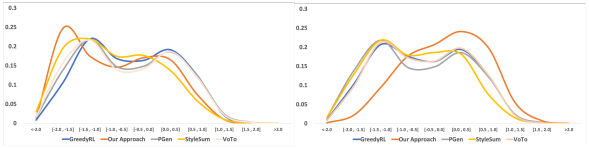
We next implement the use of single vocabulary tokens from Fan et al. (2017) in the same manner as **SymVoTo** except with the usage of one token instead of two. We refer to this method as **VoTo**.

In order to show the benefit of incorporating formality directly into the generation process, we also implement a style-transfer pipeline where we first summarize the input article and then transfer its formality to the desired level. For this purpose, we leverage the sequence to sequence implementation from Jhamtani et al. (2017) and train it on GYAFC parallel formality dataset (Rao and Tetreault, 2018). We build two models, one for formal to informal style transfer and one for informal to formal. Using the output from **PGen** method as an input to these two models, gives us the corresponding informal and formal summaries. We refer to this approach as **StyleSum**.

Next, we compare our method against the vanilla RL baseline, adapted from Paulus et al. (2018). The implementation here is similar to **ZeroRL** (Section 6.1) but with the usage of greedy baseline rewards instead of 0. We refer to this approach as **GreedyRL**.

Table 3 summarizes the results of our experiment for generating informal and formal summaries. First, we observe that as we generate more formal and informal variants, they deviate from the ground-truth summaries at lexical level, as visible in the decreasing ROUGE scores. As we later show (Section 7) through our human evaluation, this lexical difference does not affect the performance of our approach in comparison to other baselines in terms of their correctness, meaning and suitability. Secondly, we observe the desired shift in average formality scores. For both the variants, our approach better captures formality over the baseline methods. While the average formality is still on the negative spectrum for formal summaries, our method is better able to capture the oracle as compared to other baseline approaches.

The **StyleSum** method, although produces formality-diverse summaries, it fails to preserve the content of the input article, as visible by huge decline in ROUGE. This behaviour can be attributed to only a 40% overlap between the vocabulary on which the summarization and style transfer modules were trained. One of the main disadvantages for such an approach is the lack of availability of parallel corpora with the same vocabulary, for both summarization and style transfer-indicating the challenges in cascading such models and curating such corpora for these tasks.



(a) Informal summaries (b) Formal summaries

Figure 2: Distribution of formality scores of the generated summaries. Y-Axis: Fraction of datapoints, X-Axis: Intervals of formality scores.

We compare the distributions of above approaches in Figure 2. For visualization, we divide the formality score from -2.0 (more informal) to 2.0 (more formal) into 10 buckets and plot the fraction of test data points falling into these bins. The desired shifts in the distributions are visible for generating both formal and informal variants, being more profound for our approach.

Metric	Description
Formality	How formal is the given summary
Meaning Similarity	How close or similar is the meaning of the given summary with respect to the reference (ground-truth) summary
Semantic Correctness	How correct is the information present in the summary with respect to the given input article
Suitability	How well the summary suits the input article, how well it captures the key idea behind it

Table 4: Metrics considered for the qualitative analysis of the summaries generated by our approach.

7 Qualitative Evaluation

The automated evaluation is limited to comparing the summaries to a single ground-truth summary based on ROUGE metric. Hence, we further performed a crowd-sourced experiment to evaluate the quality of the generated summaries while also evaluating their formality. We compare the summaries generated by our model with those generated by **VoTo** and **GreedyRL** baseline methods. We did not consider the **Pgen** and **StyleSum** for this comparison since the former only generates a single, generic summary and the latter deviates too

Method	Informal Summaries (in %)				Formal Summaries (in %)			
	Formality	Meaning Similarity	Semantic Correctness	Suitability	Formality	Meaning Similarity	Semantic Correctness	Suitability
	All instances							
VoTo	40	62	54	52	52	56	66	54
GreedyRL	52	54	66	62	62	66	66	54
	CNN/DM instances							
VoTo	37.5	70.8	45.8	54.2	54.1	58.3	66.7	50
GreedyRL	54.2	54.2	79.2	58.3	62.5	66.7	58.3	54.2
	TL;DR instances							
VoTo	42.3	53.8	61.5	50	50	53.8	65.4	57.7
GreedyRL	50	53.8	53.8	65.4	61.5	65.4	73.1	53.8

Table 5: Percentage improvement of the proposed approach w.r.t. baseline methods for formal and informal summary generation. Each value indicates the %age of cases where the summary by our approach was rated equal to or higher in comparison to the baseline summary. For example, in 62% of the cases (All instances), the informal summary generated by our method was rated as being closer to the reference summary in meaning (**Meaning Similarity**) with respect to the summary generated by **VoTo** method. For Informal summaries, lower score on formality is desirable and for all other comparisons, a higher score is more desirable. For all the metrics, our method either performs at par or outperforms the two baselines.

much from the content in the article, as depicted by the ROUGE scores. Table 4 describes the metrics on which we perform the comparison.

The crowd-sourced experiment is conducted via Amazon Mechanical Turk[3]. 50 samples were randomly chosen from our test data, with 24 coming from CNN/DM dataset and 26 coming from Reddit dataset. The annotators were asked to rate the summaries on a discrete scale of 1 to 5 for all our requested metrics. To avoid any inter-annotator bias, we get every annotator to rate all variants of the summary generated for each test case. In total, the summaries for each sample were rated by 5 annotators. Our comparisons between any two summary variants for the same article are based on the majority opinion of these 5 annotators. To ensure the quality of the annotations, we also ask the annotators to mark all the summaries which they saw during the survey. We reject all those assignments where this question was answered incorrectly and those with less than 150 seconds work time.

Intra-Model Comparison: The annotators rate the formality of a given summary, with 1 being the least and 5, most formal. We perform a comparative analysis between the formal and informal summaries generated by the same model. For our approach, the formal summary was rated as more formal in comparison to it's informal counterpart in 76% of the cases. For **VoTo** and **GreedyRL** method, this number drops down to 66%. Being consistent with the average formality scores in Table 3, this shows that our approach is able to produce better formality-diverse summaries.

Inter-Model Comparison: In order to measure the quality of our generated summaries, we also compare them with baseline outputs on Meaning Similarity, Semantic Correctness, and Suitability (Table 4), all being key requirements in any summarization system. We compare the (in)formal summaries generated by our system with the corresponding (in)formal summaries generated by the two baseline systems. For all these metrics, higher values are more desirable for both formal and informal variants. However, for comparison on Formality, when comparing informal summaries, lesser values are desirable and while comparing formal summaries, higher are more desirable. The results of our comparative study for these metrics are presented in Table 5. All the values represent the percentage of cases where our summary is rated to be better than the corresponding baseline summary by the majority of annotators. We also report the values for each dataset separately. While our method does show a decline in ROUGE scores in comparison to these methods (Table 3), probably due to diversion from the ground-truth summaries, this decline does not translate to the quality metrics in our human evaluation. We observe that our summaries either perform at par or outperform the baseline summaries on all four metrics. We conclude that our approach produces better formality-diverse summaries, while still surpassing other methods on summarization quality.

8 Conclusion

We presented a framework to generate formality-tailored abstractive summaries for a given input article. Our approach employs reinforcement learning to train the model with formality feedback on both ground-truth and sampled summaries to-

[3]https://www.mturk.com/

gether. Automatic and human evaluations show that although we observe some deviation from the ground-truth summaries with respect to baseline methods, the approach is effective in generating formality-diverse summaries while still preserving the meaning, semantic correctness and suitability. Given a suitable oracle, the proposed methodology can be easily extended to other psycho-linguistic preferences such as politeness. We plan to perform this incorporation of other such preferences that can arise in textual content.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Siddhartha Banerjee, Prasenjit Mitra, and Kazunari Sugiyama. 2015. Multi-document abstractive summarization using ilp based multi-sentence compression. In *IJCAI*, pages 1208–1214.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*.

Julian Brooke and Graeme Hirst. 2014. Supervised ranking of co-occurrence profiles for acquisition of continuous lexical attributes. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2172–2183.

Julian Brooke, Tong Wang, and Graeme Hirst. 2010. Automatic acquisition of lexical formality. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 90–98. Association for Computational Linguistics.

Niyati Chhaya, Kushal Chawla, Tanya Goyal, Projjal Chanda, and Jaya Singh. 2018. Frustrated, polite or formal: Quantifying feelings and tone in emails. In *Second Workshop on Computational Modeling of Peoples Opinions, Personality, and Emotions in Social Media, NAACL HLT*.

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *HLT-NAACL*, pages 93–98.

Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. *arXiv preprint arXiv:1306.6078*.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *arXiv preprint arXiv:1711.05217*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104.

Katja Filippova. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 322–330.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Mengqiao Han, Ou Wu, and Zhendong Niu. 2017. Unsupervised automatic text style transfer using lstm. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 281–292. Springer.

Samuel I Hayakawa and Eugene Ehrlich. 1994. *Choose the right word*. Harper Perennial.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*, pages 1587–1596.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence to sequence models. In *Proceedings of the Workshop on Stylistic Variation*, pages 10–19.

Kundan Krishna and Balaji Vasan Srinivasan. 2018. Generating topic-oriented summaries using neural attention. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1697–1705.

Shibamouli Lahiri. 2015. SQUINKY! A Corpus of Sentence-level Formality, Informativeness, and Implicature. *CoRR*, abs/1506.02306.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *The 20th SIGNLL Conference on Computational Natural Language Learning*.

Xing Niu, Marianna Martindale, and Marine Carpuat. 2017. A study of style in machine translation: Controlling the formality of machine translation output. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2814–2819.

Xing Niu, Sudha Rao, and Marine Carpuat. 2018. Multi-task neural models for translating between styles within and across languages. *arXiv preprint arXiv:1806.04357*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics*, 4:61–74.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 129–140.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, volume 1, page 3.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Conference on Empirical Methods in Natural Language Processing*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Controlling politeness in neural machine translation via side constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 35–40.

Fadi Abu Sheikha and Diana Inkpen. 2011. Generation of formal and informal sentences. In *Proceedings of the 13th European Workshop on Natural Language Generation*.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in Neural Information Processing Systems*, pages 6833–6844.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Alexey Tikhonov and Ivan P Yamshchikov. 2018. What is wrong with style transfer for texts? *arXiv preprint arXiv:1808.04365*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Michael Völske, Martin Potthast, Shahbaz Syed, and Benno Stein. 2017. Tl; dr: Mining reddit to learn automatic summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 59–63.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *ACL (1)*, pages 1395–1405.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Do Massively Pretrained Language Models Make Better Storytellers?

Abigail See, Aneesh Pappu*, Rohun Saxena*, Akhila Yerukola*, Christopher D. Manning
Stanford University
{abisee,apappu,rohun,akhilay,manning}@cs.stanford.edu

Abstract

Large neural language models trained on massive amounts of text have emerged as a formidable strategy for Natural Language Understanding tasks. However, the strength of these models as Natural Language *Generators* is less clear. Though anecdotal evidence suggests that these models generate better quality text, there has been no detailed study characterizing their generation abilities. In this work, we compare the performance of an extensively pretrained model, OpenAI GPT2-117 (Radford et al., 2019), to a state-of-the-art neural story generation model (Fan et al., 2018). By evaluating the generated text across a wide variety of automatic metrics, we characterize the ways in which pretrained models do, and do not, make better storytellers. We find that although GPT2-117 conditions more strongly on context, is more sensitive to ordering of events, and uses more unusual words, it is just as likely to produce repetitive and under-diverse text when using likelihood-maximizing decoding algorithms.

1 Introduction

In 2018, large-scale neural models such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2019) and OpenAI GPT (Radford et al., 2018) emerged as a dominant approach in NLP. By pretraining on massive amounts of unlabeled text (often orders of magnitude larger than the the target task's labeled dataset), these models achieve state-of-the-art performance across a variety of Natural Language Understanding benchmarks. In particular, the OpenAI GPT2 language model (Radford et al., 2019) achieves state-of-the-art performance on several language modeling benchmarks, even in a zero-shot setting. While GPT2's performance as a language model is undeniable, its performance as a text generator is much less clear. Though the model has generated certain impressive samples of text – such as a widely-circulated passage about Ovid's Unicorn (Radford et al., 2019) – there has been no detailed study to formalize these observations.

In this work, we perform an in-depth study of the properties of text generated by GPT2-117 (the smallest version of GPT2) in the context of story generation. By comparing to a state-of-the-art, specialized-architecture neural story generation model (Fan et al., 2018), we ask the following questions. In what ways does a large amount of open-domain pretraining data change the characteristics of generated text? In what ways does it make no difference? And is a task-specific architecture necessary?

For any probabilistic language model, the generated text is strongly affected by the choice of decoding algorithm – this is especially true for *open-ended* text generation tasks such as storytelling and chitchat dialogue (Kulikov et al., 2018; Holtzman et al., 2019). Nevertheless, most natural language generation papers evaluate only *one* decoding algorithm – this is often due to the time and expense required for human evaluation. For example, Fan et al. use top-k sampling (a decoding algorithm in which k governs the quality-diversity tradeoff), but only evaluate one value of k. However, evaluating one k gives an incomplete view of the generation system – several researchers have emphasized the importance of evaluating generation systems over the entire quality-diversity spectrum, rather than a single point on it (Caccia et al., 2018; Hashimoto et al., 2019).

In this study, we prioritize evaluating text across the *whole* k spectrum, and measuring *many* different automatic metrics, rather than a few human metrics. Though the lack of human evaluation limits our ability to measure overall quality (Liu et al., 2016; Novikova et al., 2017; Hashimoto

*equal contribution

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 843–861
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

et al., 2019), we are able to produce an objectively defined, richly detailed and reproducible evaluation of the generated text. To our knowledge, this work is the first comprehensive analysis of the characteristics of GPT2-generated text. Our study provides insight into the effect of large-scale pretraining on open-ended natural language generation, as well as the effect of k on text generated with top-k sampling. We hope our results will inform other researchers' choice of models, pretraining schemes, and decoding algorithms – decisions that can often feel like blind choices. To enable readers to browse the generated text, conduct their own evaluations, or run our evaluations on their own text, we publicly release our generated stories and evaluation code.[1]

2 Background

WritingPrompts dataset WritingPrompts (Fan et al., 2018) is a story generation dataset containing 303,358 human-written (*prompt*, *story*) pairs collected from the /r/WritingPrompts subreddit – a forum where Reddit users compose short stories inspired by other users' prompts. An example can be seen at the top of Table 2. The mean prompt length is 28.4 words and the mean story length is 734.5 words. The dataset is 887MB of text in total, contains 200 million story words, and is divided into 90% train, 5% validation and 5% test splits.

The Fusion Model The Fusion Model is a state-of-the-art neural story generation architecture trained on the WritingPrompts dataset (Fan et al., 2018). It is based on the Convolutional Seq2seq model of Gehring et al. (2017) and aims to improve two aspects of story generation: modeling long-range context and increasing relevance of the story to the prompt. To achieve the former, the model uses a multi-scale gated self-attention mechanism. For the latter, the model uses a fusion mechanism (Sriram et al., 2018) in which one seq2seq model is trained on the task, then frozen, and a second seq2seq model is trained on the task with access to the first model's hidden states. Compared to the Convolutional Seq2seq model and other baselines, the Fusion Model achieves improved perplexity, story-prompt relevance and human preference scores. The Fusion Model has a vocabulary of 104,960 words, a 3-layer encoder and 8-layer decoder in the first seq2seq model, and a 5-layer encoder and 5-layer decoder in the second model – in total, 255.4 million parameters.

GPT2-117 GPT2 (Radford et al., 2019) is a large Transformer language model trained on WebText, a diverse corpus of internet text (not publicly released) containing over 8 million documents equalling 40GB of text in total. The full-size GPT2 model, which has 1542 million parameters, obtains state-of-the-art results on a variety of language modeling and other Natural Language Understanding benchmarks. At the time of our experiments, Radford et al. had only released the smallest of the models, known as GPT2-117.[2] This model, which we use for our experiments, has 12 layers and 117 million parameters. Like the full-size GPT2 model, it has a vocabulary of 50,257 byte-pair-encoding (BPE) tokens. The BPE encoding allows the model to encode and generate any Unicode string, regardless of preprocessing, tokenization, or vocabulary size. The model has a context size of 1024, meaning it can process text up to 1024 BPE tokens in length.

Decoding algorithms Inspired by Neural Machine Translation, most early attempts at open-ended neural text generation (such as conversational response generation) used the *beam search* decoding algorithm (Shang et al., 2015; Serban et al., 2016). Like greedy decoding, beam search is a *likelihood-maximizing* decoding algorithm – given the input sequence x, the objective is to find an output sequence y which maximizes $P(y|x)$. However, researchers have shown that for open-ended generation tasks (including storytelling), beam search produces repetitive, generic and degenerate text (Holtzman et al., 2019).

More recently, *top-k sampling* has emerged as a primary decoding algorithm for open-ended text generation (Fan et al., 2018; Radford et al., 2019). In top-k sampling, on each step of the decoder the probability distribution over the vocabulary is truncated to the top k tokens, then re-normalized. The next token is sampled from the new distribution. Top-k sampling can be regarded as somewhere between a likelihood maximizing algorithm (when $k = 1$; greedy decoding) and an unbiased sampling algorithm (when $k =$ vocabulary size). Fan et al. use top-k sampling (with $k = 10$) to

[1] Code and generated stories available at https://github.com/abisee/story-generation-eval

[2] Since conducting our experiments, larger models have been publicly released. At the time of writing, the full-size GPT2 model has not been publicly released.

generate stories, and Radford et al. show impressive samples of generated text (primarily from the full-size GPT2 model) for $k = 40$.

3 Experimental Details

Preprocessing Fan et al. truncate WritingPrompts stories to 1000 words before training and testing. Due to the limited context size of GPT2-117, we additionally exclude (*prompt*, *story*) examples that are longer than 1024 BPE tokens when concatenated. The resulting dataset, which we call WritingPrompts-1024, has 192,364 training, 11,115 validation, and 10,686 test examples.

The Fusion Model We use the pretrained version of the Fusion Model, which is available in the Fairseq framework (Ott et al., 2019). For comparability with GPT2-117, we evaluate the Fusion Model on WritingPrompts-1024 (see Table 1), obtaining perplexities similar to those reported by Fan et al. on the full WritingPrompts dataset.

GPT2-117 In order for the model to condition on prompts and generate stylistically correct stories, we finetune GPT2-117 on WritingPrompts-1024.[3] We frame WritingPrompts as a language modeling task, representing the prompt and story as a single sequence separated by a delimiter token. We finetune the pretrained model until convergence using the default hyperparameters provided in the HuggingFace repository (though we reduce batch size to fit on a single GPU), and use the finetuned model for all further evaluations.

We compute the *word-level* perplexity of the finetuned GPT2-117 on the WritingPrompts-1024 dataset. That is, we normalize the total negative log probability of the target text by the number of *word*-level (i.e. Fusion Model) tokens, not the number of BPE tokens. This enables us to compare the perplexities of the two models, despite the tokenization difference (Radford et al., 2019). The finetuned GPT2-117 obtains a test set word-perplexity of 31.54[4] – six points lower than the Fusion Model.

Generation settings For both models, we generate stories using top-k sampling, obtaining 1000 stories (from 1000 different test set prompts) for several values of k ranging from 1 to vocabulary size. We use softmax temperature 1. Like Fan et al., we generate exactly 150-word stories and block the Fusion Model from generating <UNK>.

To obtain human-written stories for comparison, we truncate WritingPrompts-1024 test set stories to 150 words (discarding those shorter than 150 words). To reduce variance, measurements for human stories are computed over this entire set (rather than just 1000 stories).

Model	Valid ppl	Test ppl
Fusion Model	37.05	37.54
GPT2-117	31.13	31.54

Table 1: Word-level perplexities on WritingPrompts-1024 for the Fusion Model and finetuned GPT2-117.

4 Story-prompt relatedness

Prior research has observed that seq2seq systems frequently produce text that is unrelated to the provided context – particularly under likelihood-maximizing decoding algorithms such as beam search. The issue has inspired multiple explanations (Jiang and de Rijke, 2018) and multiple solutions – such as alternative training objectives (Li et al., 2016), decoding objectives (Baheti et al., 2018; See et al., 2019), and architectural changes (Fan et al., 2018). In this section, we measure how strongly the models condition on the prompt.

Prompt ranking accuracy For both models, we compute *prompt ranking accuracy* (Fan et al., 2018), which measures the language model's sensitivity to the provided prompt. Following the methodology of Fan et al., we randomly select 1000 human-written stories from the test set, and measure the probability (according to the model) of each story conditioned on 10 different prompts – the true prompt, plus nine randomly selected prompts. The prompt ranking accuracy of a model is the percentage of cases in which the model assigns a higher probability to the story under its true prompt than under all of the other nine. We find that GPT2-117 scores **80.16%** on this task, while the Fusion Model scores **39.8%**.[5] Random chance scores 10%. This striking result indicates

[3] We use the PyTorch re-implementation of GPT2-117 available at https://github.com/huggingface/pytorch-transformers

[4] This is similar to other GPT2-117 WritingPrompts finetuning experiments (Mao et al., 2019; Ziegler et al., 2019).

[5] Fan et al. (2018) report a prompt ranking accuracy of 16.3% for the Fusion Model. We provided the authors with our prompt ranking accuracy code (which was built on top of the authors' code). The authors indicated that the discrepancy may be due to some code version changes between the time of their original experiments and their code release.

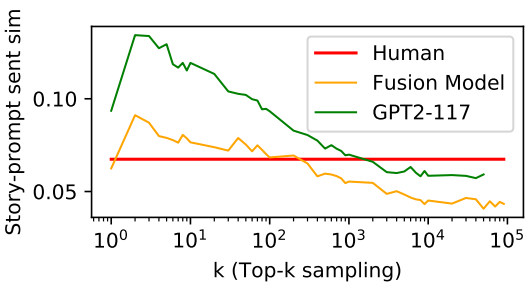


Figure 1: Compared to the Fusion Model, GPT2-117 produces stories that are more semantically similar to the prompt. Similarity decreases as k increases.

that GPT2-117 conditions on the prompt much more strongly than the Fusion Model. This is notable, especially because the fusion technique is intended to improve story-prompt relevance.

N-gram similarity For $n = 1, 2, 3$, we measure the percentage of generated n-grams that also appear in the prompt. For all n and k, we find that GPT2-117 has a higher overlap (i.e. copies more from the prompt) than the Fusion Model – see Figure 6 in the Appendix. Furthermore, for $k < 100$, the GPT2-117 overlap is generally much higher than human levels. Both these phenomena can be seen in Table 2, where, for $k = 10$, GPT2-117 copies words such as *queen* more often than both the Fusion Model and the human-written story.

Sentence embedding similarity To capture a higher-level notion of semantic similarity, we measure *story-prompt sentence similarity* – the cosine similarity of story-prompt sentence pairs, averaged by taking the mean over all pairs. Sentences are represented by the embedding method of Arora et al. (2017) – a weighted average of the GloVe embeddings (Pennington et al., 2014) of the words, with the first principal component removed. As shown in Figure 1, we find a similar pattern as for n-gram similarity: GPT2-117 generates sentences that are more similar to the prompt than the Fusion Model for all k, and both models' prompt similarity decreases as k increases.

Named entity usage Generally, most named entities mentioned in the prompt (such as *Queen* and *England* in Table 2), should also be mentioned in the story. Using the spaCy named entity recognizer,[6] we measure the *prompt entity usage rate*, which is the percentage of all prompt named entities that appear in the story.[7] As shown in Figure 7 in the Appendix, we find that GPT2-117 uses more of the prompt named entities than the Fusion Model (as well as more named entities overall), but both models use fewer named entities than humans when k is less than vocabulary size.

These patterns can be seen in Table 2: GPT2-117 uses the prompt entities *Queen* and *England* whereas the Fusion Model does not (for either k), and GPT2-117 uses specific time entities such as *Thursday* and *3:26 PM*. While the human story introduces highly-related entities such as *Charles Windsor* and *Prince of Wales* that were not in the prompt, neither model does this (for either k).

Conclusion In this section, we found that GPT2-117 conditions on the prompt much more strongly than the Fusion Model – a result which holds both in language modeling and generation settings. The latter result supports Radford et al.'s informal observation that GPT2 has a 'chameleon-like' ability to 'adapt to the style and content of the conditioning text'.[8] We speculate that GPT2-117's stronger conditioning ability may derive from its Transformer decoder architecture, whose powerful self-attention is used for story-prompt attention. Though the Fusion Model uses a similar self-attention mechanism in the decoder (i.e., story side), the prompt-story attention has a simpler formulation – for example, there are no separate key and value vectors (Gehring et al., 2017). Lastly, we note that very strong prompt-conditioning is not always a good thing – GPT2-117 often generates stories that copy too much or too literally from the prompt when k is small (this can be seen in Figure 6 in the Appendix).

5 Coherence

A good story generation model should produce coherent text with a logical ordering of events. Similarly, the underlying language model should be a good coherence scorer – assigning higher probability to coherent text than incoherent text. Barzilay and Lapata (2008) evaluate a coherence scorer by measuring its ability to rank shuffled human-written text as less coherent than the original unshuffled text. We use this method to evaluate our story generation models.

[6] https://spacy.io

[7] Given that we limit stories to 150 words, this percentage is lower than it would be if we generated longer stories.

[8] https://openai.com/blog/better-language-models/

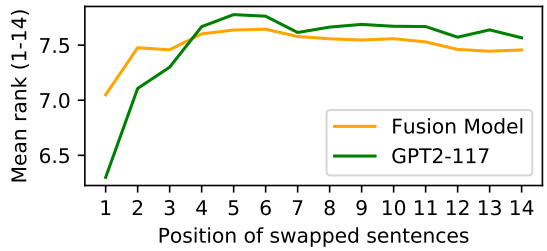


Figure 2: Sensitivity of the models to swapped sentences in different positions. A higher mean rank indicates higher sensitivity (i.e. the model assigns lower probability) relative to other positions. Both models are less sensitive to swapped sentences at the beginning of the text, compared to later. GPT2-117 shows this pattern more strongly, indicating greater use of context.

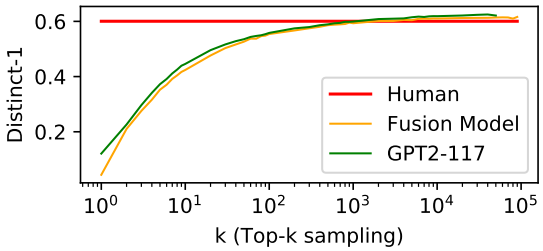


Figure 3: Repetition (low distinct-1) is primarily caused by choice of decoding algorithm (here low k), not insufficient training data. GPT2-117 is trained on $45\times$ more data than the Fusion Model, but is similarly repetitive for all k.

For each story in the test set, we select the first 15 sentences. We then produce 14 corrupted versions of the story by switching each pair of adjacent sentences. We use the language model to compute the probability of each of the 14 corrupted stories, as well as the original story. The model's error rate is the percentage of cases in which it rates any of the 14 corrupted candidates better than the original candidate. Random guessing yields 93.33% error. Both models perform well on this task – the Fusion Model has an error rate of **3.44%** and GPT2-117 an error rate of **2.17%**. This 36.92% error reduction indicates that GPT2-117 is more sensitive to ordering of events.

We also investigate how the position of the swap affects its plausibility (relative to other positions). Figure 2 shows, for each swap position, the mean rank assigned to that swap by the model (where rank 1 is the most probable of the 14 corrupted candidates, and rank 14 the least probable). GPT2-117 assigns a much lower rank to the first few swap positions (i.e., rates them more probable) than the later positions. The Fusion Model shows a similar but less pronounced pattern. This shows that both models are less sensitive to out-of-order sentences that occur at the beginning of the text, than those occurring later.[9] The stronger pattern for GPT2-117 may be due to its stronger context conditioning (as shown in Section 4) – thus becoming more sensitive as context increases. However, even for the first three swaps, GPT2-117 is more accurate than the Fusion Model at distinguishing the swapped text from the original.

[9]It's also possible that out-of-order sentences are inherently harder to detect at the beginning of text.

6 Repetition and rareness

Generic, under-diverse and repetitive text is a well-documented problem in neural text generation (Jiang and de Rijke, 2018). While there are many proposed solutions to the problem (Li and Jurafsky, 2016; Vijayakumar et al., 2018; Baheti et al., 2018; Zhang et al., 2018; See et al., 2019), it has been shown that a primary cause is likelihood-maximizing decoding algorithms such as greedy decoding and beam search (Holtzman et al., 2019). In this section we investigate the role of large-scale pretraining, and the role of k, in this problem.

N-gram repetition The *distinct-n* metric of a piece of text is the number of unique n-grams divided by the total number of generated n-grams (Li et al., 2016). We measure distinct-n of the generated stories for $n = 1, 2, 3$. A high ratio indicates a high level of within-story lexical diversity, while a low ratio indicates a large amount of within-story repetition. As shown in Figure 3, both models' unigram diversity is far below that of human text when k is small. For example, at $k = 10$ (the setting used by Fan et al.), the Fusion Model obtains a distinct-1 of 42.4%; much less than the human level of 60.0%. This results in a high level of repetition, as shown in Table 2: for $k = 10$, both models repeat many phrases (such as *always*, *so scared*, and *finally*).

For bigrams and trigrams, the pattern is similar to unigrams (see Figure 9 in the Appendix). For both models, distinct-n increases as k increases, converging to a value close to the human level as k approaches vocabulary size. Though GPT2-117 has a slightly higher distinct-n than the Fusion Model for most values of k, the difference is negligible compared to the influence of k. We

make three conclusions from these patterns: (1) Our findings support Holtzman et al.'s observation that repetition is strongly related to choice of decoding algorithm, and that likelihood maximizing algorithms (such as top-k sampling with low k) are a primary cause of repetition. (2) The models have in fact learned the correct rate of repetition in human text – they are able to match this rate when they sample from their full (untruncated) distribution. (3) Repetition is unlikely to be solved by more pretraining data alone – even though GPT2-117 is trained on 45 times as much data as the Fusion Model, it produces text that is almost equally repetitive (for equal k).

Rare word usage We compute the *mean log unigram probability* of the words in the generated story[10] – a high value indicates using fewer rare words while a low value indicates more rare words. As shown in Figure 12 in the Appendix, word rareness is primarily governed by k – however, GPT2-117 has a lower mean log unigram probability (i.e., uses more rare words) than the Fusion Model for all equal values of $k \geq 2$. This can be seen for example in Table 2, where GPT2-117 generates rarer words such as *idle* and *copious* for $k = 1000$. GPT2-117 also generates fewer stopwords than the Fusion Model, for all equal k.

GPT2-117's slightly higher rare word usage (compared to the Fusion Model) might be explained by: (1) its BPE encoding, which allows it to generate new words, not just those in a fixed vocabulary; (2) pretraining on a large amount of diverse text, allowing it to learn to produce a greater variety of words; (3) stronger conditioning on the prompt as described in Section 4 – which may inject more rareness into the generated text.

Conclusion Choice of decoding algorithm is a primary factor in diversity and repetition problems, with likelihood-maximizing algorithms the main culprit. Although GPT2-117 generates more rare words and is very slightly less repetitive than the Fusion Model, the difference is small compared to the effect of k, indicating that training data alone is unlikely to solve these problems.

7 Syntactic style and complexity

A well-trained story generation model should match both the syntactic style and complexity of its training data. Low complexity can be a sign of less sophisticated writing, while high complexity can be a sign of poor readability (Beers and Nagy, 2009; McNamara et al., 2010). In this section, we measure some features related to the syntactic style and complexity of the generated stories.

Sentence length Sentence length is a simple but effective feature to estimate readability and syntactic complexity of text (Kincaid et al., 1975; Roemmele et al., 2017). We find that both models generate sentences that are on average shorter than human sentences when k is small, but converge to approximately human length as k increases (see Figure 8 in the Appendix).

Part-of-speech usage It has been shown that the distribution of parts-of-speech (POS), and more generally the distribution of POS n-grams[11] is a useful feature to represent the style of a piece of text (Argamon et al., 1998; Ireland and Pennebaker, 2010; Roemmele et al., 2017).

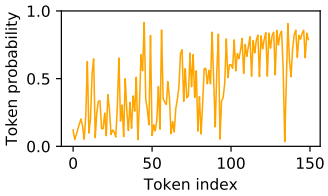
Firstly, we compare the part-of-speech distributions of the model-generated text and the human text (see Figure 11 in the Appendix). Both models (especially GPT2-117) closely fit the human POS distribution as k approaches vocabulary size.[12] This implies that, as with lexical diversity, the models have no difficulty fitting the statistical distribution of human syntax. However, under likelihood-maximizing decoding algorithms such as low k, a completely different distribution emerges, in which text contains more verbs and pronouns than human text, and fewer nouns, adjectives and proper nouns.

Secondly, we measure the syntactic diversity of the text using the distinct-n metric for POS n-grams ($n = 1, 2, 3$) – see Figure 10 in the Appendix. As with lexical diversity (see Section 6), we find that syntactic diversity is similar for the two models, is very low when k is small, and matches human level as k approaches vocabulary size. It's likely that for low k, the syntactic under-diversity of the text is largely caused by lexical under-diversity (i.e. repetition). However, we note that as k increases, lexical diversity reaches human level sooner than syntactic diversity – for example, GPT2-117's lexical distinct-3 reaches human level at $k = 600$ (Figure 9c), but its POS distinct-

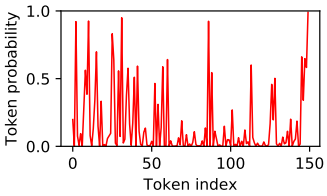
[10] The unigram probability distribution was calculated with respect to the WritingPrompts training set.

[11] For example, the sentence *I like cats* has the POS bigrams PRONOUN VERB and VERB NOUN.

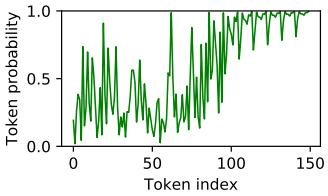
[12] One exception is Proper Noun: both models fail to produce enough of these even as k approaches vocabulary size.



(a) **Fusion Model** ($k = 2$): *I had never seen a man so young before. I had never seen him before, but he had always seemed to be a man of a man. He was young, and he was young. He was a man of a man, and a man who was young, and a man who was* [...]



(b) **Human Text**: *"Looks like the rain's stopped." I peered out the window. Art was right; time to get to work. "Alright, let's move out." I could hear the scraping of the stone armor as the men slowly stood. Despite the training,* [...]



(c) **GPT2-117** ($k = 2$): *I've always been a man of the people. I've always been a strong man. I've always been a strong man. I was born in the city, I was raised in the country. I was raised in a family that wasn't very good. I 'm not a good man.* [...]

Figure 4: Under top-k sampling with small k ($k = 2$), the two models (left and right) produce text that falls into increasingly confident repeating loops. By contrast, human text (center) maintains an irregular pattern of surprising (low probability) tokens. The human text probabilities are measured with respect to the Fusion Model, but similar patterns hold for GPT2-117. Inspired by Holtzman et al. 2019's figure showing probabilities under beam search.

3 reaches human level at $k = 6000$ (Figure 10c). This implies that, even when the text is no more repetitive than human text, it may still be syntactically repetitive (using the same part-of-speech patterns repeatedly).

Conclusion We find when k is small, syntactic complexity of generated text is low, consisting of shorter sentences and a narrower range of syntactic patterns. However, as k approaches vocabulary size, the syntactic style of generated text closely matches human syntactic patterns. As with n-gram diversity in Section 6, our results show that syntactic under-diversity is primarily caused by low k, not insufficient training data.

8 The element of surprise

Model confidence over time Several researchers have observed that *model over-confidence* (the model placing high probability on a small range of tokens) can cause poor quality generation (Jiang and de Rijke, 2018; Holtzman et al., 2019). In particular, they show that for likelihood-maximizing decoding algorithms such as beam search, model confidence can increase in a snowball-like effect, getting stuck in a loop of repetitive but increasingly self-confident text. We observe this problem in both our models when k is small. For example, in Figure 4, both models fall into self-reinforcing repetitive loops with rising confidence. The loop is difficult to break – the Fusion Model briefly escapes (shown as a sudden downwards spike), but quickly returns. By contrast, the human text does not show a strong rising trend in probability, and intermittently uses low probability words throughout.[13]

We formalize these anecdotal observations by measuring the average probability of each of the first 150 word-level tokens in the story (Figure 5). We find that even when teacher-forcing on human text, the token probabilities increase slightly as the story progresses. This is likely due to the usefulness of additional context, which increases the model's prediction accuracy. By comparison, we find that when generating with top-k sampling, the probabilities increase more rapidly, and the increase is even more rapid for smaller k. This confirms that likelihood-maximizing decoding algorithms (such as top-k sampling with small k) lead to more rapidly increasing model over-confidence. Furthermore, we find this pattern holds for both models, with probabilities increasing at a similar rate for equal k. This indicates that, like repetition, model over-confidence is unlikely to be solved by more training data, and is largely governed by choice of k.

Overall model confidence We also measure the models' overall confidence, as represented by the total log probability (according to the model) of the generated story. For both models, we find that story probability decreases as k increases – see Figure 13 in the Appendix. This makes sense, as higher k means sampling tokens with lower probability. As k approaches the vocabulary size, the Fusion Model's generated story

[13] Gehrmann et al. (2019) also identify presence of low probability words as an indicator of human-generated text.

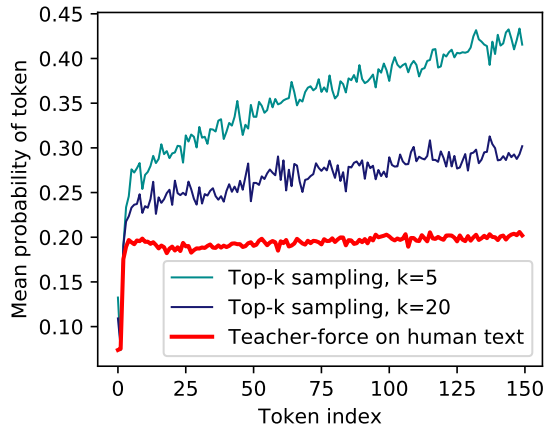


Figure 5: Mean probability for each of the first 150 word-level story tokens. When teacher-forcing the model on human text, probability increases slowly. When generating with top-k sampling, probability increases faster, especially for smaller k. This plot is for the Fusion Model; similar patterns hold for GPT2-117.

probability matches the probability it assigns to human-written WritingPrompts stories. Interestingly however, the same is not true for GPT2-117, which converges to a story probability that is *lower* than the probability it assigns the human stories. This means that under full (non-truncated) sampling, the Fusion Model produces text that is *equally surprising* (to itself) as the WritingPrompts stories, whereas GPT2-117 produces text that is *more surprising* to itself. Explaining this observation is an open question – we speculate that GPT2-117's WebText pretraining may cause it to generate (under high k) text in a style or genre that is less predictable than WritingPrompts stories.

9 Concreteness

Brysbaert et al. (2014) define the *concreteness* of a word as 'the degree to which the concept denoted by a word refers to a perceptible entity'. Concrete words are generally easier to remember than abstract words, and psycholinguists have theorized they may be learned differently (i.e., concrete words by direct experience and abstract words by text and discourse). Brysbaert et al. provide human concreteness ratings for 40,000 common English lemmas rated on a scale from 1 to 5.[14] We use these ratings to measure the mean concreteness of the nouns and verbs in the story text – see Figure 14 in the Appendix.

We find that, for the same k, GPT2-117 tends to generate more concrete words than the Fusion Model, and that for both models, concreteness converges to approximately human levels as k increases. Interestingly, however, when k is small, the noun concreteness is much *higher* than human levels, whereas the verb concreteness is much *lower*. This indicates that for small k, both models produce stories that, compared to human-written stories, have too many physical objects (as opposed to abstract nouns), and too few physical actions (as opposed to abstract verbs). This reflects the trend demonstrated in Table 2: when k is small, the models tend to generate descriptive sentences with mostly *is* verbs (e.g. *I was always so excited*), and physical nouns (e.g. *mother, father, queen*). Only when k increases do we see more tangible actions (e.g. *The bar patrons snickered*) and abstract nouns (e.g. *pain, glances*). A detailed example, with all nouns and verbs annotated with concreteness, is in the Appendix (Table 3).

10 Conclusions

The effect of massive pretraining In this study, we find that GPT2-117 is a better story generation model than the Fusion Model in several specific ways: it conditions much more strongly on the provided context, is more sensitive to correct ordering of events, and generates text that is more contentful (using more rare words, concrete words, and named entities). In particular, the stronger conditioning result is striking, as the Fusion Model is a complex task-specific architecture designed to increase story-prompt relevance. This demonstrates that sometimes, a general-purpose model architecture can outperform a complex task-specific architecture when provided with enough pretraining data.

However, we find that in other aspects, GPT2-117 performs *no better* than the Fusion Model: when k is small, the models generate text that is equally lexically under-diverse, syntactically under-complex, and repetitive – with a tendency to fall into a snowball effect of increasing overconfidence. However, these problems correct themselves (i.e., the metrics match human levels) when the models generate from their untruncated distribution. Our results show that these oft-cited neural generation problems are *not* the fault of the models themselves (which are in fact statisti-

[14] For example, the nouns *television*, *darkness*, and *idea* are rated 4.83, 3.85 and 1.61 respectively, and the verbs *talk*, *see*, and *hope* are rated 4.07, 3.21 and 1.25 respectively.

Prompt: You just discovered England's biggest secret: every time they say long live the queen it extends the queens life.	
Human-Written Story: I am immortal, I have inside me blood of kings. My name in Charles Windsor and I am heir to a throne. Prince of Wales, they've called me that my entire life and now I lay on my death bed and my mother - still going strong. I'm not sure how it's possible but she seems ageless, almost timeless. My whole life I've waited for that [...]	
Fusion Model ($k = 10$): *I have no idea when my father would come to visit this place, I was always so excited. I was always so scared. I had to go back to work. I was a little scared of him, but when I was younger, he had never come in for a visit. His mother told me, "I'm so scared", as he always did. She never talked to me about anything* [...]	**GPT2-117** ($k = 10$): ** " So what are we going to do? " * It was the first time she asked me this question that I ever had asked her. It has taken years but finally I am finally able to do it. It seems that this queen is the queen of England and she's finally the one to tell me that this queen is the reason why she's not being allowed to die.* [...]
Fusion Model ($k = 1000$): *"Where am I? What happened?" "Having been working on my job for over 6 hours now, I do not know how you worked!" "I have been working for the last three years. Surely I am an ancient god now." The bar patrons snickered. "Hello?" "Those last three years have been worse than a year ago." Pain.* [...]	**GPT2-117** ($k = 1000$): *It was an odd occasion for the Queen of England to meet with her. The plane sat idle at 3:26 PM on a Thursday night. Yesterday, the Queen had taken it upon herself to try and get a good look at the plane which had recently been found abandoned. A copious amount of curious glances from around the room until* [...]

Table 2: A prompt and human story from the dataset, plus the models' top-k generated stories, for two values of k.

cally well-trained to match human text for these metrics), nor caused by too little training data (as these problems are not improved by GPT2-117's extensive pretraining). Instead, they are primarily caused by likelihood-maximizing decoding algorithms – such as greedy decoding, beam search, and top-k sampling with low k.

The effect of k This study detailed the typical characteristics of long-form text generated by neural language models in open-ended settings, under both high entropy (large k) and low entropy (small k) decoding algorithms. The negative characteristics of low k output (genericness, repetition, oversimplicity) are by now familiar to researchers. However, we also uncovered some less obvious characteristics of low-k generated text: compared to human-written text, it tends to copy more from the provided context (particularly GPT2-117); it contains more verbs and pronouns but fewer nouns and adjectives; its nouns are more concrete but its verbs are less concrete; and it uses a smaller range of syntactic patterns (a phenomenon that can't be entirely attributed to n-gram repetition).

As k increases to vocabulary size, we find that the model-generated text closely fits the human text on most of the metrics we measured. However, it is clear by inspection that the high-k model-generated text lacks many crucial aspects such as commonsense reasoning, world knowledge and multi-sentence coherence – an example of this superficially fluent but nonsensical text can be seen in Table 4 in the Appendix. We believe that true progress in open-ended Natural Language Generation will come from attempting to address these high k problems – i.e., strategies to imbue the language model with better reasoning, knowledge and planning abilities – rather than continuing to seek ways to mitigate the diversity and repetition problems of the low k setting.

Limitations of this study This study uses only the smallest version of GPT2. It is likely that the larger versions of GPT2 may exhibit stronger statistical differences for the metrics we examine. Such a study would illustrate the effect of larger model capacity, and more fully reveal the possible benefits of massive pretraining. We release our annotation code so that other researchers may repeat our study on more models and datasets.

This study did not include human evaluation, which is currently the only reliable way to assess overall text quality, as well as quantify the deficiencies of high k output described above (coherence, reasoning, and world knowledge). As such, this study quantifies the *diversity* side more than the *quality* side of the quality-diversity tradeoff. Consequently, this study demonstrates the importance of developing better methods to computationally quantify notions such as text coherence, logicality and commonsense correctness – an effort that may ultimately hold the key to generating text with those desirable attributes.

11 Acknowledgments

This work was funded by the Gerald J. Lieberman Fellowship, Tencent, and the DARPA CwC program under ARO prime contract no. W911NF-15-1-0462. We also thank the reviewers for their helpful comments.

References

Shlomo Argamon, Moshe Koppel, and Galit Avneri. 1998. Routing documents according to style. In *First International workshop on innovative information systems*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *Empirical Methods in Natural Language Processing*.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Scott F Beers and William E Nagy. 2009. Syntactic complexity as a predictor of adolescent writing quality: Which measures? which genre? *Reading and Writing*, 22(2):185–200.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior Research Methods*, 46(3):904–911.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language gans falling short. In *NeurIPS Workshop on Critiquing and Correcting Trends in Machine Learning*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Association for Computational Linguistics*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. Gltr: Statistical detection and visualization of generated text. *arXiv preprint arXiv:1906.04043*.

Tatsunori Hashimoto, Hugh Zhang, and Percy Liang. 2019. Unifying human and statistical evaluation for natural language generation. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Molly E Ireland and James W Pennebaker. 2010. Language style matching in writing: Synchrony in essays, correspondence, and poetry. *Journal of personality and social psychology*, 99(3):549.

Shaojie Jiang and Maarten de Rijke. 2018. Why are sequence-to-sequence models so dull? Understanding the low-diversity problem of chatbots. In *EMNLP Workshop on Search-Oriented Conversational AI*.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, Fog count and Flesch reading ease formula) for navy enlisted personnel.

Ilya Kulikov, Alexander H Miller, Kyunghyun Cho, and Jason Weston. 2018. Importance of a search strategy in neural dialogue modelling. *arXiv preprint arXiv:1811.00907*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372*.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Empirical Methods in Natural Language Processing*.

Huanru Henry Mao, Bodhisattwa Prasad Majumder, Julian McAuley, and Garrison W. Cottrell. 2019. Improving neural story generation by targeted common sense grounding. In *Empirical Methods in Natural Language Processing*.

Danielle S McNamara, Scott A Crossley, and Philip M McCarthy. 2010. Linguistic features of writing quality. *Written communication*, 27(1):57–86.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Empirical Methods in Natural Language Processing*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. Fairseq: A fast, extensible toolkit for sequence modeling. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI tech report*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI tech report*.

Melissa Roemmele, Andrew S Gordon, and Reid Swanson. 2017. Evaluating story generation systems using automated linguistic analyses. In *KDD Workshop on Machine Learning for Creativity*.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? how controllable attributes affect human judgments. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI Conference on Artificial Intelligence*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Association for Computational Linguistics*.

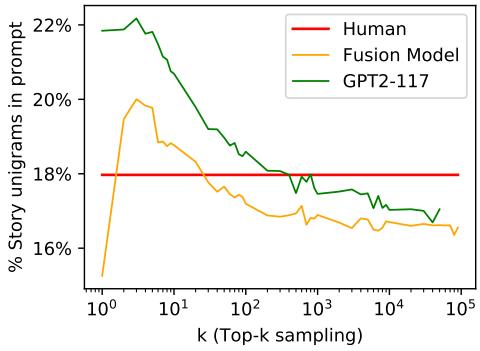
Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2018. Cold fusion: Training seq2seq models together with language models. In *Proc. Interspeech 2018*, pages 387–391.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search: Decoding diverse solutions from neural sequence models. In *AAAI Conference on Artificial Intelligence*.

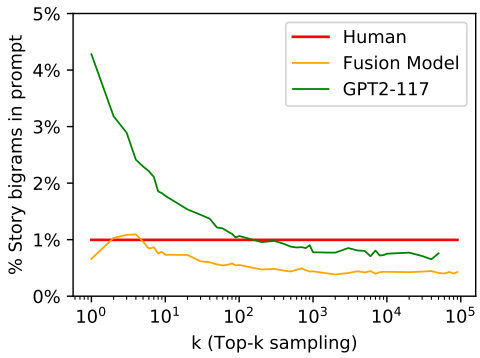
Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2018. Learning to control the specificity in neural response generation. In *Association for Computational Linguistics*.

Zachary M Ziegler, Luke Melas-Kyriazi, Sebastian Gehrmann, and Alexander M Rush. 2019. Encoder-agnostic adaptation for conditional language generation. *arXiv preprint arXiv:1908.06938*.

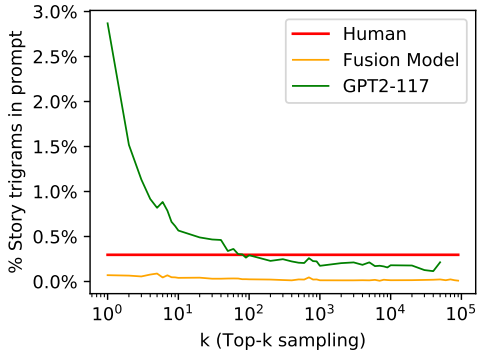
Appendix



(a) Percent of all story unigrams that are in the prompt.

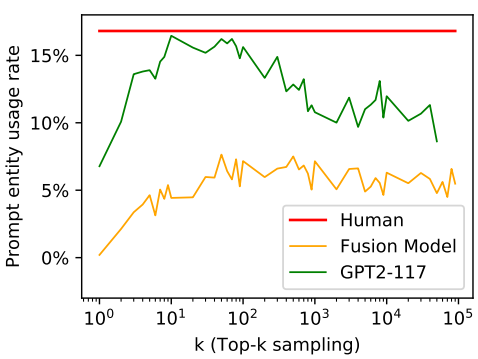


(b) Percent of all story bigrams that are in the prompt.

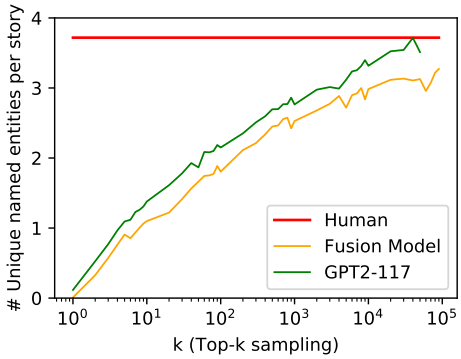


(c) Percent of all story trigrams that are in the prompt.

Figure 6: n-gram similarity between prompt and story, for $n = 1, 2, 3$, for both models and all k. GPT2-117 copies many more n-grams from the prompt than the Fusion Model. See Section 4 for discussion.



(a) The proportion of all prompt named entities that are used in the story.



(b) The number of unique named entities that appear in the story.

Figure 7: Prompt entity usage rate (left) and mean number of unique named entities in the story (right), for both models and all k. GPT2-117 generally uses a larger proportion of the prompt named entities, and more named entities overall, than the Fusion Model. Both models generally use fewer named entities than human text when k is less than vocabulary size. See Section 4 for discussion.

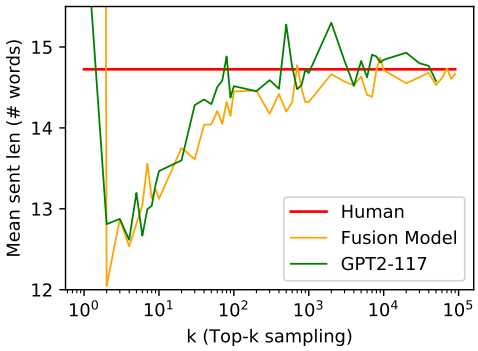
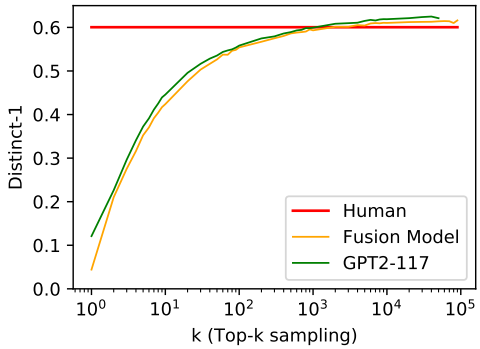
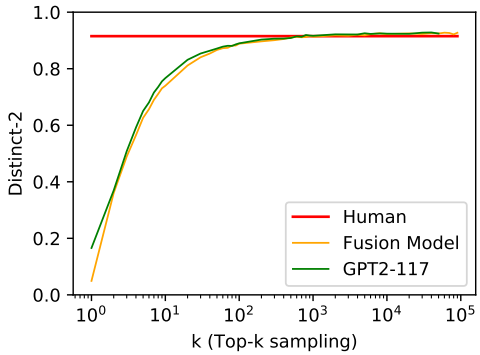


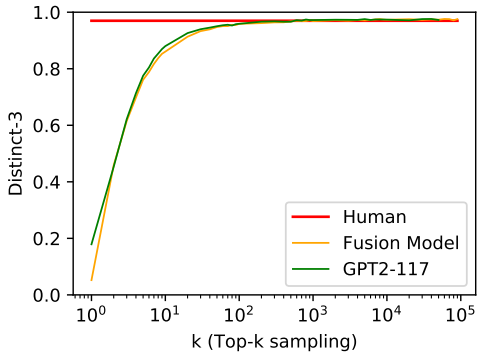
Figure 8: Mean sentence length for both models and all k. For both models, sentence length increases as k increases. The spike at $k = 1$ is due to long repeating sequences with no sentence-ending token. See Section 7 for discussion.



(a) Distinct-1 (ratio of unique unigrams in the story to total number of generated unigrams in the story).

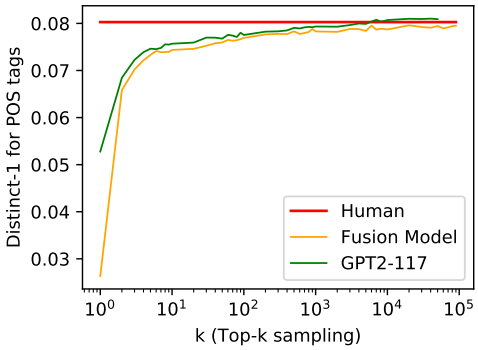


(b) Distinct-2 (ratio of unique bigrams in the story to total number of generated bigrams in the story).

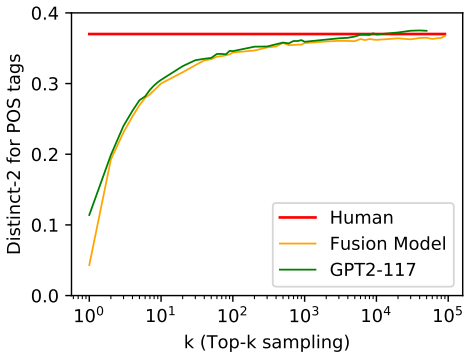


(c) Distinct-3 (ratio of unique trigrams in the story to total number of generated trigrams in the story).

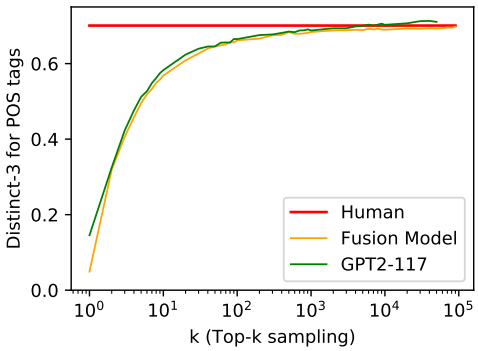
Figure 9: Distinct-n for $n = 1, 2, 3$, for both models and all k. The ratios, which represent lexical diversity, increase as k increases, with GPT2-117 reaching human levels at $k = 2000$ for unigrams, $k = 800$ for bigrams and $k = 600$ for trigrams. Lexical diversity is slightly higher for GPT2-117 than for the Fusion Model for equal k, but the primary determining factor is k. See Section 6 for discussion.



(a) POS tag distinct-1 (ratio of unique POS unigrams in the story to total number of generated POS unigrams in the story).



(b) POS tag distinct-2 (ratio of unique POS bigrams in the story to total number of generated POS bigrams in the story).



(c) POS tag distinct-3 (ratio of unique POS trigrams in the story to total number of generated POS trigrams in the story).

Figure 10: POS tag distinct-n metric for $n = 1, 2, 3$, for both models and all k. The ratios, which represent syntactic diversity, increase as k increases, with GPT2-117 reaching human levels at $k = 6000$ for unigrams, $k = 9000$ for bigrams, and $k = 6000$ for trigrams. Syntactic diversity is slightly higher for GPT2-117 than for the Fusion Model for equal k, but the primary determining factor is k. See Section 7 for discussion.

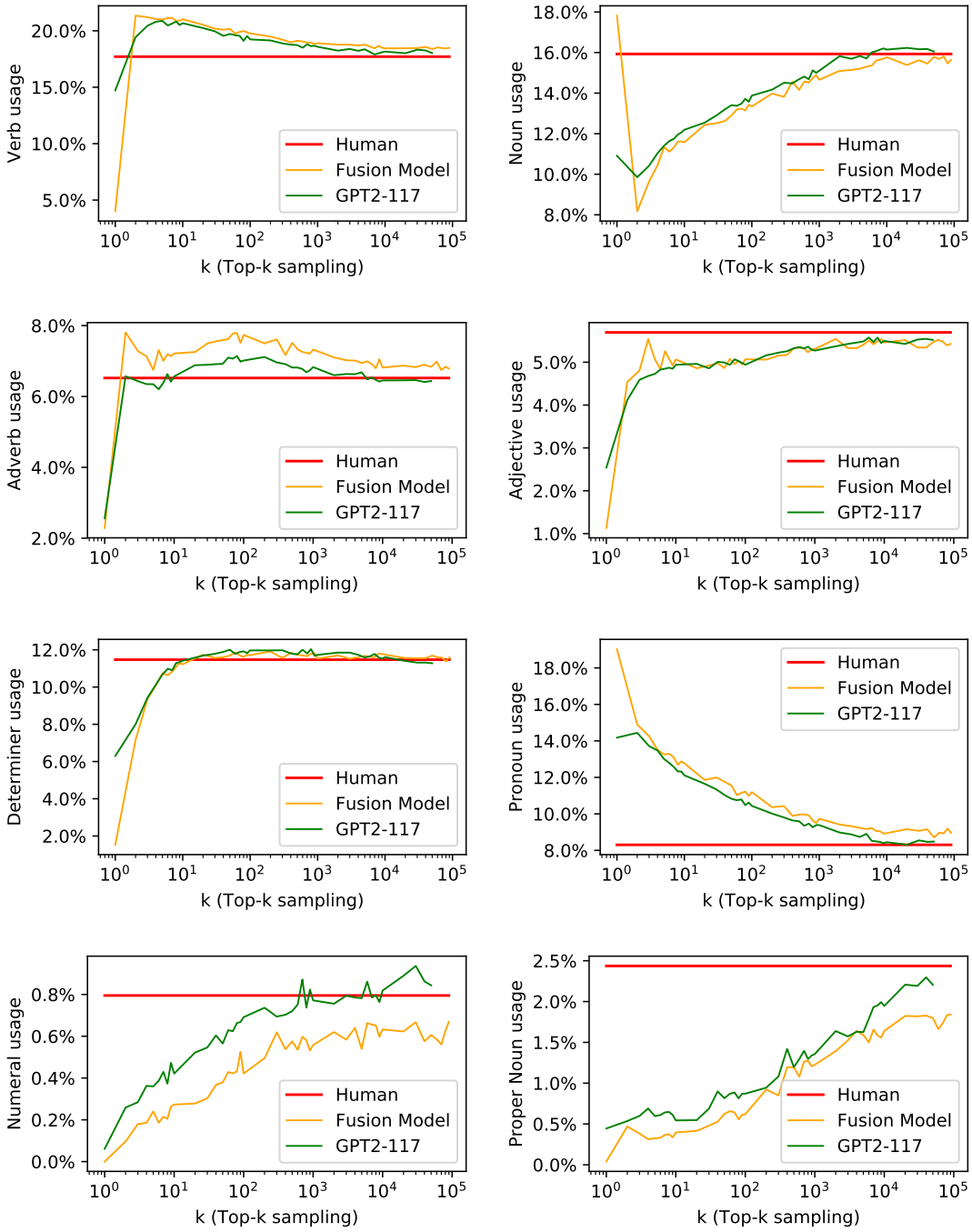
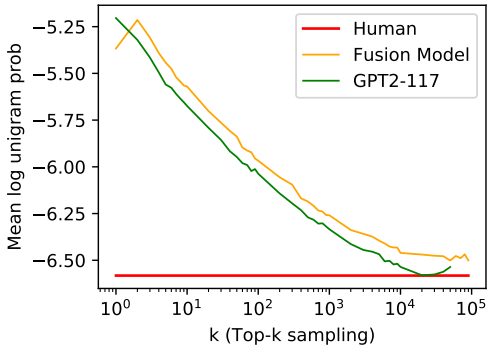
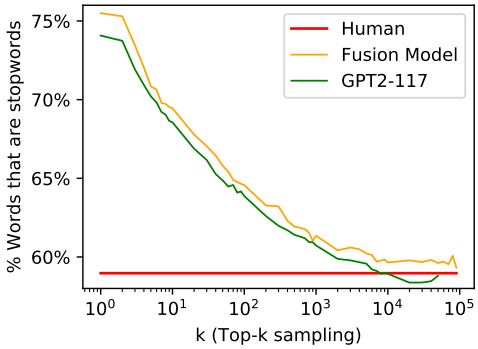


Figure 11: Usage of different POS tags in the generated stories. GPT2-117 tends to fit the human distribution more closely than the Fusion Model as k approaches vocabulary size, in particular producing more specific POS categories such as Numeral and Proper Noun. When k is small, generated text is characterized by more verbs and pronouns, and fewer nouns, adjectives, numerals and proper nouns, than human text. See Section 7 for discussion.

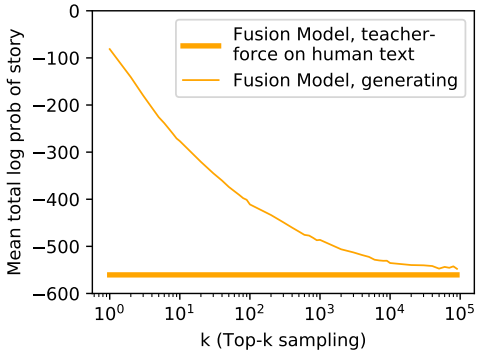


(a) The mean log unigram probability of generated words. Higher values indicate using fewer rare words while lower values indicate using more rare words.



(b) The percent of generated words that are stopwords, for both models, across different k. We use the NLTK English stopword list.

Figure 12: Rare word usage metrics for both models and all k. GPT2-117 produces slightly more rare words (left) and slightly fewer stopwords (right) than the Fusion Model, for equal values of k. These rareness metrics do not reach human levels until k is close to vocabulary size. See Section 6 for discussion.



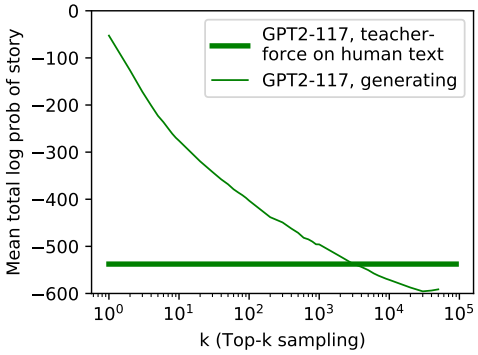


Figure 13: The mean total log probability of the story (150 words), as measured by the models on their own generated output and on human-written stories. Interestingly, the Fusion Model (left) converges to the same probability it assigns to human-written stories as k approaches vocabulary size, whereas GPT2-117 (right) converges to a lower probability. See Section 8 for discussion.

Prompt: A man finally discovers his superpower... well into his 80's.

Fusion Model ($k = 10$):

I be a child . bear into a world of darkness , and in the world , there be only a few people . My mother and I be always alone . She have never be like this . But , she have no idea what we be together to be . We be always alone . We be always together . And , of course , she have never exist . But , I never think about it , because she be always alone . In the world , it was n't like she have a power . I be always alone , and there be a time when the light would turn on . There be a time when I could see the light , and I could see it in her eyes , and I could see the light , and I could see it

Mean noun concreteness: 3.892
Mean verb concreteness: 2.173

GPT2-117 ($k = 10$):

A man in an old white coat sit in a corner. He wear a black jacket with a blue tie. His hair be grey. His eyes be black, but his voice be deep and clear. He look to where the man be sit, and he see a smile on his lips. It be a smile he know he would see from his own eyes. But he be too late. He be on the sidewalk by the river when the man come. He be wear a black coat with a purple tie. He have a black tie and a white shirt. But he be still wear a white suit. And it seem he would look back at him. A smile on his face. A look his friend do n't recognize. He have no

Mean noun concreteness: 4.720
Mean verb concreteness: 2.488

Fusion Model ($k = 1000$):

For a brief moment, the dark green of his eyes flash the last lines of reality into existence, finally fade slowly into existence so that he could not feel it. Only the millions of pixels float in his peripheral vision almost disappear. His radio respond to the pinging very hard silence of the outside world. Seven people have visit his main internal network for what seem like a lifetime but this time, the only switch to an external supply system that he could simply take advantage of. Unable to convey feelings about the last words he would have to endure, but it have respond to the innumerable messages and countless sleepless hours. Most of them be always available on its surface, just to make sure. In his quest for to spend eternity on death, he send

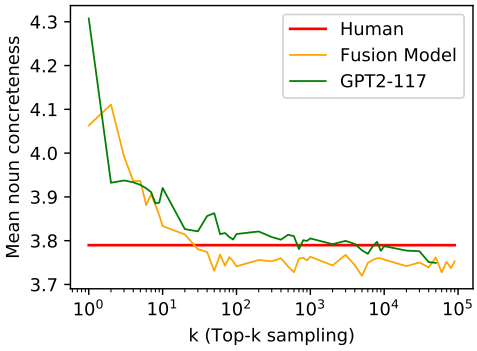
Mean noun concreteness: 3.201
Mean verb concreteness: 2.435

GPT2-117 ($k = 1000$):

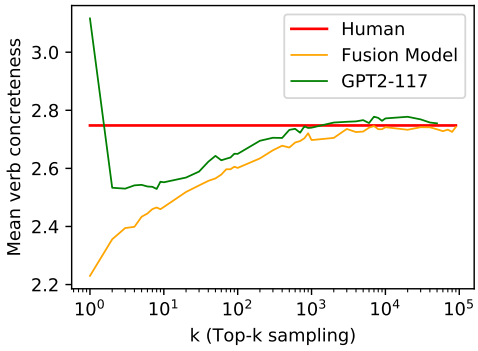
(First time poster, hope its ok) The young boy, watch tv, spot the television onscreen, before glance around to see the screen start the countdown on the tv, point to the screen in " It 's both the same. " "... let 's... let 's try this and... we will team up so that... we can... have the same power....like... so we can use this superpower over and over again. " A brief silence. Only a familiar conversation, interrupt his mad dash movement, follow with his high pitch slurred and wither voice : " I ca n't stand anyone talk like that son*s*. " More casual conversation that interrupt his childish step be rush to the scissors.

Mean noun concreteness: 3.793
Mean verb concreteness: 3.162

Table 3: Generated stories from both models, under $k = 10$ and $k = 1000$. Nouns are highlighted in green and verbs in yellow. The highlighting intensity reflects the word's concreteness rating. For equal k, GPT2-117 generally generates more concrete words than the Fusion Model. For both models, low k is characterized by high noun concreteness (e.g. physical objects such as *jacket*) and low verb concreteness (e.g. non-physical actions such as *be*). Conversely, high k is characterized by low noun concreteness (e.g. abstract concepts such as *reality*) and high verb concreteness (e.g. physical actions such as *talk*). See Section 9 for discussion.



(a) Mean concreteness rating (1-5) of nouns in the story.



(b) Mean concreteness rating (1-5) of verbs in the story.

Figure 14: Mean concreteness rating of the nouns and verbs in the story, for both models and all k. GPT2-117 generally produces nouns and verbs that are more concrete than the Fusion Model for the same k. For both models, as k increases, noun concreteness reduces and verb concreteness increases. See Section 9 for discussion.

Prompt: *In an alternative reality where sleep is non-existent among living beings, our protagonist (spontaneously or after an event) falls asleep in which he/she experiences for the first time in human history what a dream is.*
GPT2-117 ($k = 1000$)**:** *I sat in my bed as my girlfriend sat behind me, buzzing into her e-reader, letting the day's stories write themselves on her's monitor. Like a blur, all the usual high-asyllabic drivel and senseless ramblings that normally attracted the attention of a horrid sleeping creature huddled about me like a faucet. She did not know how I placed this car with her. But I tried, first tried to ignore that I had hired the services of a dog to help and then quietly used it in a desperate bid to drive the car through the lawn. Each and every day, I watched her drool down an old dusty hardwood mattress her beady eyes trying desperately to think of this rotting dream.* [...]

Table 4: An example of syntactically natural but nonsensical text, generated with high k. Though the text is fluent and readable, it is surreal and bizarre. See Section 10 for discussion.

Self-Adaptive Scaling Approach for Learnable Residual Structure

Fenglin Liu[1], Meng Gao[3], Yuanxin Liu[4,5] and Kai Lei[2]*
[1]ADSPLAB, School of ECE, Peking University, Shenzhen, China
[2]ICNLAB, School of ECE, Peking University, Shenzhen, China
[3]School of ICE, Beijing University of Posts and Telecommunications, Beijing, China
[4]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[5]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
fenglinliu98@pku.edu.cn, gaomeng@bupt.edu.cn
liuyuanxin@iie.ac.cn, leik@pkusz.edu.cn

Abstract

Residual has been widely applied to build deep neural networks with enhanced feature propagation and improved accuracy. In the literature, multiple variants of residual structure are proposed. However, most of them are manually designed for particular tasks and datasets and the combination of existing residual structures has not been well studied. In this work, we propose the Self-Adaptive Scaling (*SAS*) approach that automatically learns the design of residual structure from data. The proposed approach makes the best of various residual structures, resulting in a general architecture covering several existing ones. In this manner, we construct a learnable residual structure which can be easily integrated into a wide range of residual-based models. We evaluate our approach on various tasks concerning different modalities, including machine translation (IWSLT-2015 EN-VI and WMT-2014 EN-DE, EN-FR), image classification (CIFAR-10 and CIFAR-100), and image captioning (MSCOCO). Empirical results show that the proposed approach consistently improves the residual-based models and exhibits desirable generalization ability. In particular, by incorporating the proposed approach to the Transformer model, we establish new state-of-the-arts on the IWSLT-2015 EN-VI low-resource machine translation dataset.

1 Introduction

Recently, residual learning attracts considerable attention in training deep neural networks, and many efforts have been devoted to study the utilization of residual structure in tasks across a broad span of fields, including but not limited to computer vision (He et al., 2016a; Huang et al., 2017; He et al., 2016b; Szegedy et al., 2017) and natural language processing (Vaswani et al., 2017; Devlin et al., 2019). Residual structure, which alleviates the so-called gradient exploding or vanishing problem in optimization (He et al., 2016a), enables the training of neural networks with great depth by building skip connections between layers.

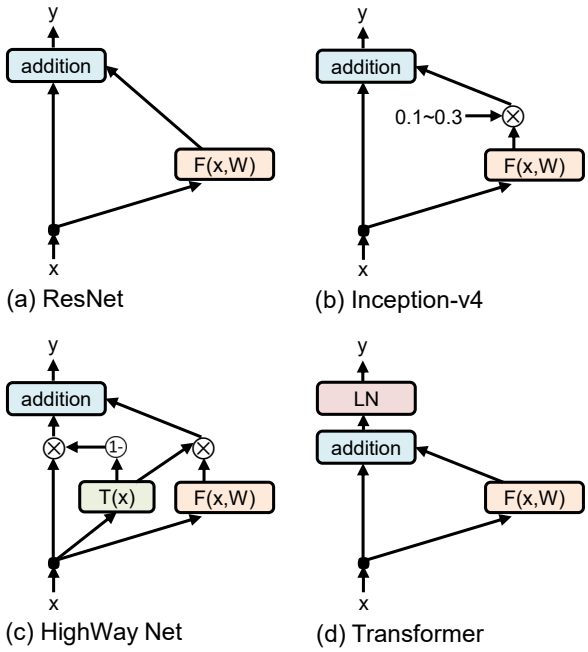


Figure 1: Various types of residual structures: (a) ResNet (He et al., 2016a); (b) Inception Net (Szegedy et al., 2017); (c) Highway Net (Srivastava et al., 2015); (d) Transformer (Vaswani et al., 2017), where LN represents layer normalization (Ba et al., 2016).

Generally, the residual structures (as illustrated in Figure 1) can be formulated as:

$$\boldsymbol{y} = \mathcal{G}(\alpha \cdot \boldsymbol{x} + \beta \cdot \mathcal{F}(\boldsymbol{x}, \mathcal{W})) \quad (1)$$

where $\boldsymbol{x}$ denotes the input (i.e., the skip connection), $\mathcal{F}$ denotes the residual function (i.e., residual branch) parameterized by W, and $\boldsymbol{y}$ is the output of the residual block. The balance between $\boldsymbol{x}$ and $\mathcal{F}$ is governed by the weights α and β, followed by $\mathcal{G}$, which could be either identity mapping or

*Corresponding Author

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 862–870
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

normalization.

Previous works on residual structure designing, which differ in the way that the information flows are regulated, mainly concern two elements, namely the mapping formulation (weight assignment) and the normalization mechanism. As shown in Figure 1, ResNet (He et al., 2016a), Inception-v4 (Szegedy et al., 2017) and Highway Net (Srivastava et al., 2015) explored the question on how should the residual connection be incorporated into the existing neural network structures so that the best improvements can be achieved. Recently, Transformer (Vaswani et al., 2017) applied the layer normalization (Ba et al., 2016) to help the optimization of the non-linear transformation (i.e., the $\mathcal{F}$) to some extent. At the same time, He et al. (2016b) observed considerably worse results when they utilized batch normalization (Ioffe and Szegedy, 2015) after the residual connection, the reason for which batch normalization is less employed in the residual structure.

Despite their respective advantages and success in certain fields, we argue that the structures are only particular cases of a more general one, which necessitates further insights into possible combinations. However, the determination of an effective combination may require prior knowledge of the data distribution, which is not always available, or extensive hyper-parameter exploration, which is inefficient.

In this paper, we aim at constructing a comprehensive and flexible residual structure. To this end, we propose the Self-Adaptive Scaling approach. In the residual structure, the proposed approach automatically computes scaling factors to adjust the mapping formulation and the normalization mechanism, respectively. By assigning different importance to the skip connection, the residual branch and a normalized result, the scaling factors adaptively controls the topology of the residual building blocks.

As a result, the structure learned by our proposed approach can be easily generalized to various kinds of tasks and data, dispensing with the time-consuming architecture search, to some extent. The proposed learnable residual structure can be easily integrated into existing residual-based models. We evaluate the proposed approach on representative residual models for various tasks. The experiment results and analyses attest to our argument and the effectiveness of the proposal.

Overall, the contributions are summarized as followed:

- We proposed a novel self-adaptive scaling (*SAS*) approach to acquire a learnable residual structure, which allows deep neural models to automatically learn the residual structure and can cover different types of existing ones.
- The proposed approach is simple and can be easily applied to a wide range of existing residual-based models. According to our empirical studies, the *SAS* can enable existing models to achieve consistent performance gains, demonstrating its generalization ability to a wide range of existing systems.
- The experimental results on the IWSLT-2015 EN-VI show that *SAS* helps the Transformer-Base model to perform even better than the Transformer-Big model and, encouragingly, we establish a new state-of-the-art on this low-resource machine translation dataset.

2 Related Work

In recent years, the application of residual structure to deep neural networks has become an active research topic (He et al., 2016a; Srivastava et al., 2015; He et al., 2016b; Vaswani et al., 2017; Szegedy et al., 2017). In the studies on residual architectures, there are two problems of interest. The first is how should the information from the skip connection and the residual branch be well balanced so that the best improvements can be achieved. The second is how should the neural network with residual connections be optimized so that its representation capability could be fully mined. These two types of problems are mainly addressed by designing appropriate mapping formulation and normalization mechanism, respectively, and we refer to them as *On the Connection Problem* and *On the Optimization Problem*.

On the connection problem. There are roughly three lines of methods to control the balance in residual connections: identity mapping, constant scaling ratio and adjusted scaling ratio. He et al. (2016b) designed five types of shortcut connections and discussed the possible residual connections in detail. Based on their theory and experiments, they argued that "keeping a 'clean' information path is helpful for easing optimization". The reason is that with scaling, the gradient of the residual suffers

from the gradient exploding or vanishing problem, which hinders the deep neural network from efficient optimization. Szegedy et al. (2017) adopted constant scaling to govern the residual balance in deep inception networks, which, despite its decent performance, is relatively inflexible. Highway network (Srivastava et al., 2015) is among the very first endeavors to implant residual structures into deep neural networks. It built a highway connection from the input to the output, where a transform gate was proposed to control the balance of the skip connection $\boldsymbol{x}$ and the residual branch $\mathcal{F}(\boldsymbol{x}, \mathcal{W})$, as opposed to the identity mapping.

On the optimization problem. In the realm of computer vision, PreAct-ResNet (He et al., 2016b) demonstrated that it is helpful to apply batch normalization to $\boldsymbol{x}$, instead of $\mathcal{F}(\boldsymbol{x}, \mathcal{W})$. In other words, the batch normalization acts on the output of the previous block. For natural language processing, the popular Transformer (Vaswani et al., 2017) makes use of residual connection in conjunction with layer normalization to build the model architecture and achieves record-setting performance. Layer normalization is widely believed to be helpful for stabilizing training and facilitating convergence. According to our experiments and analyses, the layer normalization can indeed facilitate optimization and therefore improve the overall performance of the model.

Different from existing work, we summarize the combination of normalization and residual connection in existing works with a general form $\boldsymbol{y} = \alpha * \boldsymbol{x} + \beta * \mathcal{F} + \gamma * \mathrm{LN}(\boldsymbol{x} + \mathcal{F})$, where the mapping formulation and the normalization mechanism are both taken into account. By changing the scaling factors α, β and γ, the topology of the residual block can be adaptively adjusted, resulting in a learnable residual structure. The learned architecture distinguishes itself from the previous ones with generality and flexibility.

Our work is also related to the line of research on neural architecture search (Zoph and Le, 2017), where the network structure is also automatically learned by algorithm. However, neural architecture search requires sampling architecture descriptions based on predicted probability from a controller network that is optimized via reinforcement learning, which is time-consuming. But our proposed approach can be trained directly with the loss functions of different tasks.

No.	α	β	γ	Architecture
1	0	0	1	$\mathrm{LN}(\boldsymbol{x} + \mathcal{F})$ (Vaswani et al., 2017)
2	1	1	0	$\boldsymbol{x} + \mathcal{F}$ (He et al., 2016a)
3	1	β	0	$\boldsymbol{x} + \beta * \mathcal{F}$ (Szegedy et al., 2017)
4	α	1	0	$\alpha * \boldsymbol{x} + \mathcal{F}$ (He et al., 2016b)

Table 1: Four particular cases in formula (3), which cover four representative residual structures, i.e., Transformer (Vaswani et al., 2017), ResNet (He et al., 2016a), Inception-v4 (Szegedy et al., 2017) and shortcut-only gating proposed in He et al. (2016b).

3 Architecture

In this section, we first briefly introduce the Scaling Gate in Section 3.1, which is used to predict the scaling factors for the mapping formulation and the normalization mechanism. Then, based on the scaling factors, in Section 3.2, we describe how to adaptively make the best of different types of residual structure to build a learnable residual structure.

3.1 Scaling Gate

The Scaling Gate should be able to predict reasonable scaling factors. Our motivation stems from the superior performance of Feed-Forward Network used in Vaswani et al. (2017). When selecting the activation function, since we expect the final predicted value of the scaling factors to cover the range of 0∼1, we applied the Sigmoid activation function to the original outputs of the scaling gate. As a result, the scaling gate takes the $\boldsymbol{x} \in \mathbb{R}^h$ and the $\mathcal{F}(\boldsymbol{x}, \mathcal{W}) \in \mathbb{R}^h$ as input and computes the output through two linear transformations with a Tanh activation in between:

$$\mathcal{S}(\boldsymbol{x}, \mathcal{F}) = \mathrm{Tanh}([\boldsymbol{x}; \mathcal{F}]W_f + b_f)W_{ff} + b_{ff} \quad (2)$$

where [;] denotes concatenation operation, $W_f \in \mathbb{R}^{2h \times h}$ and $W_{ff} \in \mathbb{R}^{h \times 1}$ are the parameters to be learned, and $\mathcal{S}(\boldsymbol{x}, \mathcal{F})$ is followed by a Sigmoid activation function.

In HighWay Net (Srivastava et al., 2015), the inputs of the Transform Gate only involve the information of $\boldsymbol{x}$ and the structure only contains one layer of linear transformation, i.e., $\boldsymbol{x}W + b$. In contrary, we integrate the information from $\boldsymbol{x}$ and $\mathcal{F}$ and build the structure with two layers of linear transformation, which strengthens the scaling gate's expressive power. The difference is illustrated in Figure 2, and as illustrated in Table 6, our scaling gate is experimentally found to perform better.

No.	α	β	Architecture	Remarks
1	0	0	$\mathrm{LN}(\boldsymbol{x}+\mathcal{F})$	The residual structure of Transformer (Vaswani et al., 2017).
2	0	1	$\mathcal{F}$	The well-trained $\mathcal{F}$ is sufficient in representation ability.
3	1	0	$\boldsymbol{x}$	$\mathcal{F}$ is poor-trained or $\boldsymbol{x}$ is sufficient in representation ability.
4	1	1	$\boldsymbol{x}+\mathcal{F}$	The residual structure of ResNet (He et al., 2016a).
5	1	β	$\boldsymbol{x}+\beta*\mathcal{F}$	The residual structure of Inception-v4 (Szegedy et al., 2017).
6	α	1	$\alpha*\boldsymbol{x}+\mathcal{F}$	The residual structure of shortcut-only gating (He et al., 2016b).
7	1-β	β	$(1-\beta)*\boldsymbol{x}+\beta*\mathcal{F}+$ $\beta(1-\beta)*\mathrm{LN}(\boldsymbol{x}+\mathcal{F})$	The combination of Highway Net (He et al., 2016b) and Transformer.

Table 2: Seven special cases in formula (4), which include four representative structures, i.e., Transformer (Vaswani et al., 2017), ResNet (He et al., 2016a), Inception-v4 (Szegedy et al., 2017) and Highway Net (Srivastava et al., 2015).

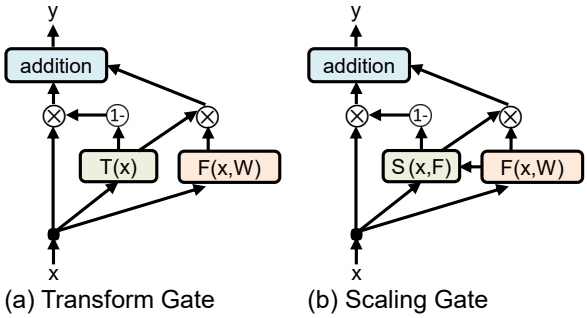


Figure 2: The difference between the Transform Gate in Highway Net (Srivastava et al., 2015) and the proposed Scaling Gate.

3.2 Self-Adaptive Scaling Approach

It is intuitive to combine different types of residual structure through adjustable scaling factors. Therefore, we reformulate the residual block as follows:

$$\boldsymbol{y} = \alpha * \boldsymbol{x} + \beta * \mathcal{F} + \gamma * \mathrm{LN}(\boldsymbol{x}+\mathcal{F}) \quad (3)$$

where α, β and γ can be predicted by scaling gates with different parameters, and LN stands for layer normalization (Ba et al., 2016).

By choosing certain values for α, β and γ, we can get several special cases, as shown in Table 1. However, from the summarized cases, we can easily find that if we set $\gamma = (1-\alpha)(1-\beta)$, the approach can also cover these four baselines. Especially, it is essential to decrease the parameters to achieve the same purpose[1]. Thus, the final self-adaptive scaling approach can be defined by the following formula:

$$\boldsymbol{y} = \alpha * \boldsymbol{x} + \beta * \mathcal{F} + (1-\alpha)(1-\beta) * \mathrm{LN}(\boldsymbol{x}+\mathcal{F}) \quad (4)$$

where α and β act as the scaling factors predicted as aforementioned. From the above formula, we can obtain seven special cases in Table 2. In all, our proposed self-adaptive scaling approach is not only the general form of several existing structures, which is able to encourage the model to take full advantage of different types of residual structures, but also gives rise to a learnable residual structure, which can be automatically learned by deep neural models from the data.

[1]The empirical results also show that $(1-\alpha)(1-\beta)$ performs better than γ (94.05 accuracy vs. 93.95 accuracy in CIFAR-10).

4 Experiment

In this section, we evaluate the proposed approach on three representative tasks in the natural language processing field, computer vision field, and cross-modal scenario, that is, image classification, machine translation and image captioning. We first briefly introduce the baseline models for comparison, the datasets, the metrics and implementation details, followed by the discussions about the experimental results. Since our major concern is the combination of different components in residual units, we keep the internal structure (i.e., the residual function $\mathcal{F}(\boldsymbol{x}, \mathcal{W})$) of each component unaffected. The training and inference strategies also remain the same as the original models. For more details, please refer to the cited publications.

4.1 Machine Translation

Baselines, Datasets, Metrics and Settings. For the task of machine translation, we adopt the popular Transformer (Vaswani et al., 2017), which is a strong baseline. The model is implemented with the code from tensor2tensor (Vaswani et al., 2018). Transformer follows the encoder-decoder paradigm, but it replaces the self-recursive operation in RNNs with the self-attention that summarizes all context. In each Transformer block, the self-attended results are post-processed by residual connection and layer normalization.

There are 133K, 4.5M, and 36M training

Method	EN-VI	EN-DE	EN-FR
Transformer-Base (Vaswani et al., 2017)			
Baseline	30.9	27.5	38.2
+ Proposal	**32.0**	**27.6**	**38.4**
Transformer-Big (Vaswani et al., 2017)			
Baseline	31.6	28.5	41.0
+ Proposal	**32.2**	**28.7**	**41.3**

Table 3: Results (BLEU) on the machine translation task. Higher means better. The proposal brings consistent and substantial improvements.

Dataset	Baseline	+ Proposal	Improvements
PreAct-ResNet (Vaswani et al., 2017)			
CIFAR-10	93.66	**94.05**	+0.39
CIFAR-100	71.29	**72.91**	+1.62

Table 4: Results (Accuracy (%)) on the image classification task, averaging over 5 runs. Higher is better. The proposal consistently outperforms the baselines as in machine translation. Especially, better improvements are achieved for models on CIFAR-100.

pairs in the IWSLT-2015 English-Vietnamese (EN-VI) (Cettolo et al., 2015), WMT-2014 English-German (EN-DE) and English-French (EN-FR), respectively. `tst2012` and `tst2013` are selected as the development and test sets, respectively, for EN-VI. For EN-DE, we use `newstest2013` and `newstest2014`; and for EN-FR, `newstest2012+2013` and `newstest2014` are selected. For experiments on the two WMT datasets, we follow the implementation settings in Vaswani et al. (2017). For experiments on the IWSLT EN-VI dataset, we set the batch size equal to 4096 and train on single GPU, as it is relatively small. For all datasets, we use a single model by averaging the last 10 checkpoints to produce the results with beam search of 4 and length penalty of 0.6.

Results. The results of machine translation are presented in Table 3. Under both the Base and the Big configuration, our proposal consistently boosts the performance of the Transformer baseline. When equipped with our proposed approach, the Transformer-Base model even transcends the big version, which is three times as large in size, on the EN-VI translation task. It shows that the scaling factors indeed help adjust the residual structure to the data distribution, which is very efficient in exploiting the expressive power of deep residual networks. Encouragingly, an union of the proposal and the Transformer-Big model achieves substantial improvement, and it outperforms the state-of-the-art method (Huang et al., 2018) in the EN-VI low resource dataset.

4.2 Image Classification

Baselines, Datasets, Metrics and Settings. In the computer vision field, we benchmark our proposed learnable residual structure with residual-based image classification systems, i.e., Pre-Activated ResNet (PreAct-ResNet) (He et al., 2016b). ResNet-110 consists of 54 double-layer residual blocks, which makes it non-trivial to optimize. To demonstrate that our *SAS* is applicable to such deep framework, we select ResNet-110 in the experiments. We retain most of the hyper-parameters in He et al. (2016b), with the exception of the weight decay rate, which is set to 0.0002, so as to guarantee more stable training. Both CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) are comprised of colored images for classification. CIFAR-100, which contains 100 classes, appears to be more difficult as compared to CIFAR-10, where there are only 10 classes. Following common practice (He et al., 2016b; Srivastava et al., 2015), accuracy rate of classification over 5 runs are reported as the evaluation results. In Srivastava et al. (2015) and He et al. (2016b), they found that it may be beneficial to attach more importance to the skip connection $\boldsymbol{x}$ for initialization, i.e., the bias term in the transform gate should be initialized with a negative value. Following this practice, in image classification task, we set the bias b_{ff} for α and β to 3 and -3, respectively, at the start of training.

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE
GLIED (Liu et al., 2019c)	80.4	-	-	**39.6**	**28.9**	58.8	129.3	**22.6**
Transformer (Vaswani et al., 2017)								
Baseline	80.2	64.9	50.7	39.0	28.4	58.6	126.3	21.7
+ Proposal	**81.2**	**65.4**	**51.2**	39.3	28.7	**59.0**	**129.8**	**22.6**
Improvements	+1.0	+0.5	+0.5	+0.3	+0.3	+0.4	+3.5	+0.9

Table 5: Performance on the MSCOCO Karpathy test split. Higher is better in all columns. The baseline enjoys a comfortable improvement with the proposed approach. Additionally, we report the performance of the recently published state-of-the-art GLIED, as we can see, our approach helps the Transformer captioning model outperforms GLIED substantially in terms of CIDEr, which further demonstrates the effectiveness of our approach.

The remaining weights are initialized in the same way as in (He et al., 2016b).

Results. As can be seen in Table 4, consistent improvements are also obtained over the baseline model, which is much deeper than the six-layer Transformer model. The increase in accuracy is 0.39 and 1.62 on the CIFAR-10 and CIFAR-100 dataset, respectively. This demonstrates that our proposal also works in deeper cases. It comes to our notice that the proposal induces better improvements on the more challenging CIFAR-100 dataset. This is presumably that the learnable structure can make the best of each component in the residual building block, which allows more flexible fitting into the multi-class image distribution, resulting in a larger space for improvement in the 100 classes scenario.

4.3 Image Captioning

Baselines, Datasets, Metrics and Settings. To further demonstrate the generalization ability of our proposed approach, we conduct experiments on the task of image captioning. The experiments are based on the multi-head attention mechanism (Vaswani et al., 2017), which has recently shown great potential and is competitive with the most advanced models (Liu et al., 2019c,b), for the reason of which we choose it as our baseline to examine the performance of our approach on the multidisciplinary task.

There are several datasets that consist of image-sentence pairs. Our reported results are evaluated on the popular Microsoft COCO (MSCOCO) (Chen et al., 2015) dataset, which contains 123,287 images. Each image in the dataset is paired with 5 sentences. The results are reported using the widely-used publicly-available splits in the work of Karpathy and Li (2015). The MSCOCO validation and test set contain 5,000 images each. Following common practice (Liu et al., 2018, 2019a), we replace caption words that occur less than 5 times in the training set with the generic unknown word token UNK, resulting in 9,567 words.

We adopt SPICE, CIDEr, BLEU, METEOR and ROUGE for testing. They are previously used as evaluation methods for image captioning, we report the results using the MSCOCO captioning evaluation toolkit (Chen et al., 2015). Among the metrics, BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) are originally designed to evaluate the performance of machine translation systems. ROUGE is widely used to examine the quality of machine-produced summaries. SPICE and CIDEr are bespoke metrics for image captioning, which measures scene graph and n-gram matching, respectively, and we refer to them as primary indicators of model performance.

Results. The results on Karpathy test split (Karpathy and Li, 2015) are reported in Table 5. By using our proposed learnable residual structure, improvements of 3.5 points and 0.9 points in terms of CIDEr and SPICE respectively can be achieved, further demonstrating the effectiveness and generalization capabilities of our approach to a wide range of tasks. More encouragingly, the proposed approach helps the baseline model achieves 129.8 CIDEr score, an improvement over GLIED (Liu et al., 2019c) by 0.5.

5 Analysis

In this section, we conduct several analyses to give further insights into our proposed approach, which are based on the image classification task and we adopt PreAct-ResNet-110 (He et al., 2016b) as the baseline model.

Analysis on Scaling Gate. Table 6 summarizes the obtained results when applying the Transform

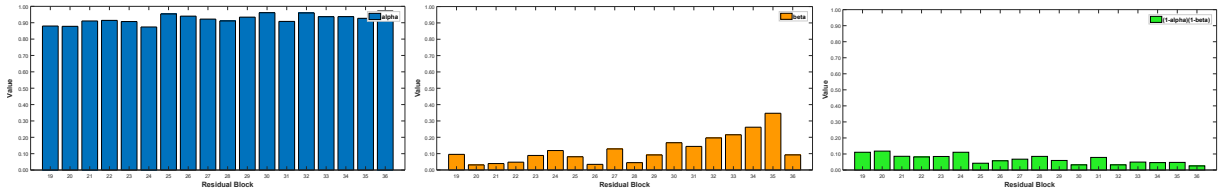


Figure 3: Illustrations of the averaged value of α (left) and β (middle), together with the corresponding $(1-\alpha)(1-\beta)$ (right), which are predicted by the proposed approach for each residual block in pretrained PreAct-ResNet-110 on CIFAR-10.

Methods	CIFAR-10
Baseline (PreAct-ResNet-110)	93.66
+ Transform Gate	92.15
+ Scaling Gate (Single Layer)	92.73
+ Scaling Gate (Full Model)	**93.82**

Table 6: The effects of applying the Transform Gate from Highway Net (Srivastava et al., 2015) and the proposed Scaling Gate to the PreAct-ResNet-110 (He et al., 2016b), where the performance is evaluated by Accuracy(%). The single layer Scaling Gate takes the form $\mathcal{S}(\boldsymbol{x}, \mathcal{F}) = [\boldsymbol{x}; \mathcal{F}]W_f + b_f$.

Architecture	Acc.(%)
$\boldsymbol{x} + \mathcal{F}$ (Baseline) (He et al., 2016b)	93.66
$\alpha * \boldsymbol{x} + \beta * \mathcal{F} + (1-\alpha)(1-\beta) * \text{BN}(\boldsymbol{x} + \mathcal{F})$	93.23
$\alpha * \boldsymbol{x} + \beta * \mathcal{F} + (1-\alpha)(1-\beta) * \text{LN}(\boldsymbol{x} + \mathcal{F})$	**94.05**

Table 7: Results on CIFAR-10 using the PreAct-ResNet-110 with the batch/layer normalization. The batch normalization is less effective than the layer normalization in residual structure.

Gate in Highway Net and our Scaling Gate to PreAct-ResNet-110, as well as the results of the vanilla model. As we can see, when equipped with Transform Gate, the effect is counter-productive on CIFAR-10 dataset. This indicates that the information from $\boldsymbol{x}$ along is not robust and effective enough to predict the scaling factors in the residual structure. The single layer version of our Scaling Gate takes into account the residual branch $\mathcal{F}$, thereby improving over the Transform Gate. It is worth mentioning that compared with the Transform Gate ($\mathcal{T}(\boldsymbol{x}) = \boldsymbol{x}W_f^T + b_f^T$), which has $(h \times h) + h$ learnable parameters, Scaling Gate (Single Layer) only introduces $(2h \times 1) + 1$ learnable parameters, which is much more efficient. By modeling the scaling factor with Scaling Gate (Full Model), a 0.16 points promotion is achieved over the baseline on CIFAR-10, which further demonstrates the advantages and effectiveness of the proposed Scaling Gate.

Analysis on Self-Adaptive Scaling. Averaging over 10,000 experimental examples, we display in Figure 3 the value of α, β and $(1-\alpha)(1-\beta)$ in each residual block of the pretrained PreAct-ResNet-110 on CIFAR-10. The filter sizes for the blocks at the bottom, middle and top of the model are different. We only show the representative blocks at the middle of the model due to space limitation. The first column shows that in almost all cases, α is greater than 0.9, which indicates that identity mapping is very helpful to information transfer and eases the optimization of deep neural networks. This can be attributed to the facilitated backward propagation of error signals by identity mappings. It is shown in the second column that as the number of network layers increases, the value of β grows simultaneously, which indicates that the representation ability of the residual branch $\mathcal{F}$ is stronger when it comes closer to the output of the model. The main reason is that the error signal passed to $\mathcal{F}$ is more adequate in the upper blocks, which is beneficial to optimization. As can be seen from the third column, more importance is assigned to the normalized result when it comes to the lower parts of the entire architecture, which means that the value of $(1-\alpha)(1-\beta)$ is larger. This is because that in the underlying static blocks of deep neural networks, the guidance from error signal is weak and the optimization is unstable, thus making the introduction of layer normalization necessary.

In all, the proposed approach regulates the information from individual components with scaling factors to build the learnable residual structure, which helps make the best of different type of residual structure, resulting in an effective combination for better performance.

Analysis on Using Batch Normalization. The batch normalization is used commonly in the field of computer vision. Therefore, we replace the layer normalization with the batch normalization in the proposed approach to see the difference. As shown

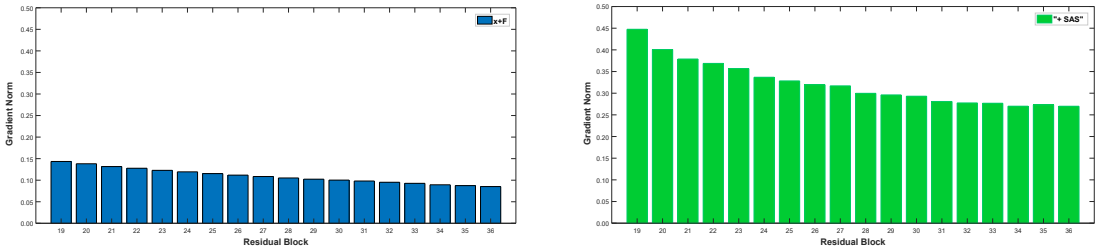


Figure 4: Gradient norm of the output of residual blocks of two different structures that are based on PreAct-ResNet-110. The values are calculated as an average of over 10,000 random examples in the training set of CIFAR-10. We conduct analyses on the blocks from the model's middle part. The *SAS* denotes the self-adaptive scaling approach.

in Table 7, applying batch normalization has a negative effect on the performance. Most importantly, it lags behind the baseline by a noticeable margin, which shows that batch normalization is less effective than layer normalization in the residual structure. We speculate that layer normalization is able to mitigate the training issue in the form of exploding gradient induced by the adjusted ratio provided by the layer normalization's own parameters, while batch normalization could not, which could be intuitively derived by the framework in Hanin and Rolnick (2018). It is probably the reason why He et al. (2016b) observed considerably worse results when they applied batch normalization on the residual structure.

Analysis on Better Optimization Capability. To understand how the proposed approach helps the optimization of deep neural models, we inspect into the gradient norm of the output of each residual block in the pre-trained PreAct-ResNet-110 on CIFAR-10. The gradients are averaged over 10,000 randomly selected training examples. As shown in the left plot of Figure 4, the gradients of the "$x + \mathcal{F}$" structure are basically the same, indicating that all the residual blocks have similar speed for gradient descent and optimization. In contrast, the right plot of Figure 4 reflects that more gradients are allocated to the lower blocks with the help of *SAS*, and the overall gradient values are greater. This phenomenon is interesting and finally gives rise to better results, as shown in Table 4. We conjecture that the layers distant from the model output cannot receive adequate guidance from the error signal, thus requiring more gradient for optimization. Moreover, since the layer normalization is able to stabilize the information flow and accelerate convergence, adaptively incorporating the residual structure with layer normalization can also facilitate optimization. By allocating more importance to layer normalization in the residual blocks via scaling factors, the layers at the bottom of the network can be better optimized, which is in line with the foregoing analysis on Self-Adaptive Scaling approach.

6 Conclusion

In this work, we focus on building a learnable residual structure, which automatically learns the design of residual structure from data, instead of the handy-crafted designs in previous work. We propose the Self-Adaptive Scaling approach to achieve this goal, which combines various residual structures via the predicted scaling factors, resulting in a general residual structure covering several existing models. The proposed approach is simple and can be easily integrated into existing residual-based models. Experiments on the machine translation, image classification and image captioning tasks validate the effectiveness of the proposed method, which successfully promotes the performance of all the strong baselines. This also demonstrates the generalization ability of our method. In particular, when being applied to the recently proposed Transformer model, our approach establishes new state-of-the-arts on the IWSLT EN-VI low resource machine translation task, which further substantiates its efficiency. Detailed analyses prove that the proposed approach can also promote the optimization ability of deep neural networks, and is conducive to exerting the expressive power of existing models.

Acknowledgments

This work was partially supported by the Shenzhen Fundamental Research Project (No. ZDSYS201802051831427). We thank all the anonymous reviewers for their constructive comments and suggestions.

References

Lei Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *ACL Workshop*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Roldano Cattoni, and Marcello Federico. 2015. The iwslt 2015 evaluation campaign. In *IWSLT 2015, International Workshop on Spoken Language Translation*.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Boris Hanin and David Rolnick. 2018. How to start training: The effect of initialization and architecture. In *NeurIPS*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016b. Identity mappings in deep residual networks. In *ECCV*.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society.

Po-Sen Huang, Chong Wang, Sitao Huang, Dengyong Zhou, and Li Deng. 2018. Towards neural phrase-based machine translation. In *ICLR*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3128–3137. IEEE Computer Society.

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical report, Computer Science Department, University of Toronto.

Fenglin Liu, Meng Gao, Tianhao Zhang, and Yuexian Zou. 2019a. Exploring semantic relationships for image captioning without parallel data. In *ICDM*.

Fenglin Liu, Yuanxin Liu, Xuancheng Ren, Xiaodong He, Kai Lei, and Xu Sun. 2019b. Aligning visual regions and textual concepts for semantic-grounded image representations. In *NeurIPS*.

Fenglin Liu, Xuancheng Ren, Yuanxin Liu, Kai Lei, and Xu Sun. 2019c. Exploring and distilling cross-modal information for image captioning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5095–5101. ijcai.org.

Fenglin Liu, Xuancheng Ren, Yuanxin Liu, Houfeng Wang, and Xu Sun. 2018. simnet: Stepwise image-topic merging network for generating detailed and comprehensive image captions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 137–149. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 311–318. ACL.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, François Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Lukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. In *AMTA*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Barret Zoph and Quoc V. Le. 2017. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

BIOfid Dataset: Publishing a German Gold Standard for Named Entity Recognition in Historical Biodiversity Literature

Sajawel Ahmed[1], Manuel Stoeckel[1], Christine Driller[2], Adrian Pachzelt[3], Alexander Mehler[1]
[1]Goethe University Frankfurt
[2]Senckenberg Nature Research Society
[3]Frankfurt University Library
{sahmed,mehler}@em.uni-frankfurt.de

Abstract

The *Specialized Information Service Biodiversity Research* (*BIOfid*) has been launched to mobilize valuable biological data from printed literature hidden in German libraries for over the past 250 years. In this project, we annotate German texts converted by OCR from historical scientific literature on the biodiversity of plants, birds, moths and butterflies. Our work enables the automatic extraction of biological information previously buried in the mass of papers and volumes. For this purpose, we generated training data for the tasks of *Named Entity Recognition* (NER) and *Taxa Recognition* (TR) in biological documents. We use this data to train a number of leading machine learning tools and create a gold standard for TR in biodiversity literature. More specifically, we perform a practical analysis of our newly generated *BIOfid dataset* through various downstream-task evaluations and establish a new state of the art for TR with 80.23% F-score. In this sense, our paper lays the foundations for future work in the field of information extraction in biology texts.

1 Introduction

Data is the gold to any machine learning (ML). Most ML approaches to *Natural Language Processing* (NLP) address modern, high-resource languages (such as English or Chinese) rather than historical, low-resource languages. As a consequence, feasible ML-tools for processing historical documents are still rare. In this paper we consider corpora of historical German texts in order to extract useful information about biological systems in the past (e.g. species, biotopes etc.).

As a contribution to closing the gap between NLP of modern and of historical languages, we present the newly annotated *BIOfid dataset* for *Named Entity Recognition* (NER) and for *Taxa Recognition* (TR) in the domain of biology, the first of its kind concerning the German language. Our approach is especially designed to address the exploration of biodiversity data[1] from historical documents. We perform a large-scale annotation of scanned texts converted by OCR from historical scientific books on the biodiversity of plants, birds, moths and butterflies, thereby creating the necessary training data to accomplish the task of biological NER and TR using various ML algorithms. Our work facilitates an automatic extraction of biological information so far buried in the bulk of papers and volumes (see Table 1). Over-

Input sentence:
Ahmed observes that Iris grows in Mai in Frankfurt.
TR output:
Ahmed observes that [Iris]TAXON grows in Mai in Frankfurt.
Biological NER output:
[Ahmed]PER observes that [Iris]TAXON grows in [Mai]TIME in [Frankfurt]LOC.

Table 1: Example for our selected tasks.

all, our newly generated dataset provides a gold standard and hereby lays the foundations for future work, such as relation extraction and classification based on extracted biological named entities and taxa.

We perform a practical analysis of our dataset via various downstream-task evaluations. First, we generate a baseline for recognizing taxonomic entities by constructing a sequence tagger based on skip-n-grams and external knowledge resources (i.e. WikiData). Secondly, we apply the best publicly available word embeddings for German and use them alongside our BIOfid dataset as an input for training high-performing neural mod-

[1]Biodiversity is the science which measures the variability and diversity of animals and plants.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 871–880
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

els for NER, namely BiLSTM, ELMo, Flair and BERT (Ahmed and Mehler, 2018; Peters et al., 2018; Akbik et al., 2018; Devlin et al., 2018). By using the optimized BiLSTM model we achieve a new best F-score of 80.23% regarding the recognition of taxonomic entities.

The remainder of the paper is organized as follows: Section 2 reviews related work. Section 3 describes the source texts and the preprocessing pipeline. Section 4 describes the annotation guidelines, process and environment for producing the BIOfid dataset, and methods (n-gram-based sequence tagger, neural models) for evaluating the practical quality of our annotated dataset. Section 5 presents the experimental results. Finally, Section 6 draws a conclusion.

2 Related Work

2018 was a vital year for the task of German NER, following a saturation period from when the last major progress was made by Lample et al. (2016). With the grammar-specific morphological processing and resource-optimization presented by Ahmed and Mehler (2018), the gap between English and German NER was closed. In the same year, with the emergence of multilingual language models such as *ELMo*, *Flair* and *BERT* (Peters et al., 2018; Akbik et al., 2018; Devlin et al., 2018), the performance of various NLP tasks, including NER, was notably improved. Hence, the task of German NER has benefited from these developments.

However, with respect to the availability of a variety of resources, there has not been much progress made until now. Regarding the standard task of NER based on four categories (`PERSON`, `LOCATION`, `ORGANIZATION`, `OTHER`), the first choice of resources for German is still the *GermEval* dataset (Benikova et al., 2014), followed by the datasets of *CoNLL* and *TüBa-D/Z* (Tjong Kim Sang and De Meulder, 2003; Telljohann et al., 2012). However, their potential for purposes outside of theoretical ML is limited. These datasets do not contain any annotations for taxonomic and temporal entities which are of key interest for biodiversity researchers.

For biological NER in the German language, there are no predecessor resources available to the knowledge of the authors; only an English counterpart exists, namely the *Copious* dataset (T.H. Nguyen et al., 2019), which has been recently published during our ongoing work. This confirms our research endeavors and shows the necessity of more data in this field. We take the English counterpart as the baseline and compare its dataset and results with our own. Overall, our work constitutes the first effort on enabling a state-of-the-art performance for neural representation learning to biological NER.

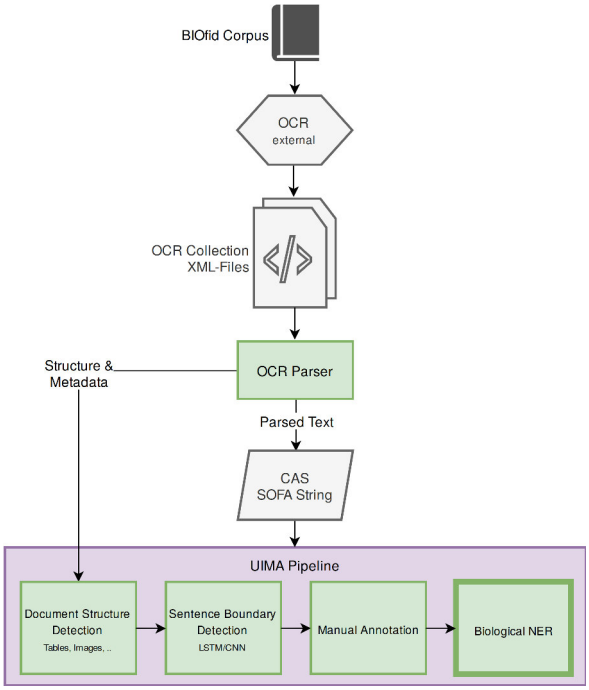


Figure 1: Flowchart showing the data cleaning steps within our preprocessing pipeline.

3 Source Texts & Preprocessing Pipeline

BIOfid Corpus The *BIOfid Corpus* is a collection of historical scientific books on central European biodiversity. It was assembled by a group of German domain experts, denoting a potential pool of relevant print-only journals and publications for historical biodiversity science. However, mainly due to license issues, not all publications could be considered for the corpus.

The available publications were scanned by an external service and subsequently paginated with the software *Visual Library*. Subsequently, every high-resolution page (400 dpi) was digitized with *ABBYY FineReader 8.0 (2005)* to ABBYY-XML, which includes structural information like paragraphs, bold/italic text, images, and table blocks.

OCR Parser The raw OCR data contained various errors, e.g. delivering typical OCR errors such as confusing letters (**ß** → **b**), or delivering

gibberish due to the wrong recognition of non-textual elements in scans such as images, figures, or tables. Furthermore, species names or their appended author citation were frequently recognized incorrectly, e.g. "Lepidium ruderale L." → "Lepidium rüderale I.".

We built the following preprocessing pipeline (see Figure 1) to clean the source data and increase its overall quality. First, the raw OCR data was passed to a parser (labeled "OCR Parser" in Figure 1). This parser read a given ABBYY-XML into a UIMA CAS, while retaining all structural information in a custom UIMA type system, which was tailored to the ABBYY-XML output.

Using a set of heuristics, the structural information was used to detect erroneous parts in the parsed text, such as page numbers, image and figure blocks mislabeled as text, text margins and table lines parsed as the characters "I" or "-", and tables containing merely non-word characters such as counts of observations[2].

The parser performed further fundamental text segmentation using the information given by the ABBYY-XML, such as tokenization and paragraph splitting. The ABBYY-XML contains tokenization information on the character basis, denoting whether a character is marking the beginning of a word. This information was used alongside plain whitespaces to tokenize the raw text, while further splitting words from non-word characters. All this information was stored in a UIMA CAS using the aforementioned type system and passed down the UIMA pipeline.

Document Structure The BIOfid corpus comprises about 15 journal titles including approximately 410 books. 201 of these books containing 969 articles were selected by domain experts as a representative sample from the entire corpus to generate training data for biological NER.

Sentence Boundary Detection In biological literature, author citations are commonly abbreviated (e.g. Carl von Linné in "Fagus sylvatica L.") as well as species names (e.g. "F. sylvatica" after the first definition). Therefore, standard rule-based tools often fail to detect the correct sentence boundaries in such unstructured raw text documents. Hence, for this task we included the LSTM-based sentence boundary detector *DeepEOS* (Schweter and Ahmed, 2019) in our preprocessing pipeline and trained it with 1,361 sentences, which were manually extracted from the BIOfid corpus. The total amount of training sentences was increased from a preliminary size of 300, since the first experimental results revealed that the SBD is crucial for the performance of our downstream-task.

[2] An example of such pages is given in Appendix C.

4 BIOfid Dataset & Methods

4.1 Annotation Guidelines

Named Entities NEs are real-world objects in a given natural language text which denote a unique individual with a *proper name* (e.g. Frankfurt, Africa, Linnaeus, BHL). This stands in contrast to the class of *common names* which refer to some kind of entities (e.g. city, continent, person, corporation) and *not* a uniquely identifiable object.

The standard task of NER focuses on the former class of proper names. However, it is often not easy to differentiate between both classes. Hence, to support the annotators in making the right decision, we created guidelines which demonstrated the rules for annotations. We gradually developed this document in collaboration with the annotators, until finalizing it as the guidelines for annotating the BIOfid corpus. The appendix shows the material which was provided to the team of annotators. First, in Appendix A some introductory examples from the BIOfid corpus are given. Next, in Appendix B the general guidelines used for producing the NER dataset are shown.

As we essentially extend the standard task of NER to our scope of biodiversity, our guidelines are built upon those used for producing the GermEval dataset (Benikova et al., 2014). For this, we take the original German text and extend it with the important adjustments described in the next paragraphs for the context of biodiversity. In contrast to Benikova et al. (2014), we do not consider derivative or partial NEs as a separate category. As the recent work of Ahmed an Mehler (2018) has shown, discarding subtle details is even beneficial, whereas fine-graded feature engineering for deep neural networks usually deteriorates the final performance.

Time In the standard task of NER, temporal information is not captured by the four base entities. However, the aspect of time is important for the research on biodiversity which is constantly evolving. Therefore, we annotated every text unit

Dataset	Sentence	PERSON	LOCATION	ORGANIZATION	OTHER	TIME	TAXON
CoNLL	18,933	*5,369*	6,579	4,441	3,968	N/A	N/A
GermEval	31,300	10,807	17,275	8,303	4,557	N/A	N/A
TüBa-D/Z	104,787	55,746	28,582	32,224	12,865	N/A	N/A
Copious	*26,277*	*2,889*	*9,921*	*N/A*	*N/A*	*2,210*	*12,227*
BIOfid	**15,833**	**5,393**	**6,785**	**1,085**	**7,849**	**5,197**	**15,085**

Table 2: Statistics for German NER datasets together with the English biological NER dataset *Copious* (T.H. Nguyen et al., 2019).

which denotes a specific temporal entity with the tag TIME (e.g. *[13.02.1835]*TIME, see more in Appendix B: Table 9). For text units which describe a time interval, we marked the starting and ending points as two distinct temporal entities.

Taxonomy Taxonomy is a field in biology that deals with the systematic classification of organisms by morphological, phenotypic, behavioral and phylogenetic characteristics. Based on a variety of common traits, a group of organisms forms a so-called taxon. A well-known example of this are the Darwin's finches, endemic birds in the Galápagos Islands. The different species (each species represents a taxon) are distinguished primarily by the size and shape of their beaks and the associated specialized diets.

Taxa are classified according to international nomenclature codes[3,4,5,6] and are delineated at different hierarchical levels, also known as taxonomic ranks. Most of us are well acquainted with the distinction between the animal and plant kingdoms, although there are other kingdoms e.g. fungi or bacteria. Subordinate to a kingdom are many more ranks such as phylum, class, order, family, genus and species. According to this, the hierarchical classification of the bird species *Struthio camelus*, the common ostrich, from the lowest to the highest taxonomic rank is as follows: *Struthio camelus* (species), *Struthio* (genus), *Struthionidae* (family), *Struthioniformes* (order), *Aves* (class), *Chordata* (phylum), *Animalia* (kingdom). Each scientific name mentioned here along with its taxonomic rank (in parentheses) represents a taxon, meaning a group of organisms with a set of common characteristics being indicative for a common ancestry.

Due to differing and evolving methods of classification, taxonomies are subject to constant change. This also applies to taxonomic nomenclature. Therefore, among others, synonymy and homonymy also play an important role in biology (e.g. there is a plant genus with the name "Paris"). The relevance of taxonomy for biodiversity research and conservation is fundamental (Thomson et al., 2018), consequently, we considered it justified to introduce the NE-category of TAXON into the process of NER.

For organisms of all taxonomic ranks, we considered scientific names (both accepted and synonyms) and vernacular names, if referring to a certain taxon, as NEs (e.g. *[Struthio camelus]*TAXON or [common ostrich]TAXON, *[Mirza zaza]*TAXON or [northern giant mouse lemur]TAXON, see more in Appendix B: Table 7). Author citation and year, usually appended to the scientific name of a taxon, were tagged as NEs of the categories PERSON and TIME, respectively (e.g. *[Falco]TAXON [Linnaeus]PER [1758]TIME*). Both author and temporal information embedded within the scientific name, were included in the NE TAXON (e.g. *[Carex praecox [Jacq.]PER var. distans]TAXON [Appel]PER*).

4.2 Annotation Process

We performed a single major series of annotations. Instead of just focusing on some inter-agreement value, we performed double checks on existing annotations on given articles through biological experts. This strategy removed the time overload associated to multi-annotations while ensuring a high quality of data.

For this scheme, a group of annotators consisting of two researchers from the project team were employed. Both researchers were native speakers of German, and, additionally had a profound background in biology. Besides, two further student assistants with similar profiles were employed to provide further assistance.

[3] http://iczn.org/code
[4] http://www.iapt-taxon.org/nomen/main.php
[5] http://www.the-icsp.org/
[6] http://talk.ictvonline.org/taxonomy/



Figure 2: Working environment for annotating the BIOfid corpus (figure taken from (Abrami et al., 2019)).

4.3 Annotation Environment

We used the *TextAnnotator* (Abrami et al., 2019), a browser-based annotation tool specifically adjusted for this project. Figure 2 shows the working environment which was provided to the annotators. On the left-hand side of the *QuickAnnotator* view, the raw OCR text from the BIOfid corpus is displayed, separated from the choice of annotation tags on the right-hand side. As sentence splitting was part of the annotation task, we did not provide a sentence view. Instead, we provided the whole article, further allowing the annotators to use contextual information while making their decisions.

4.4 Quality of Data

4.4.1 Quantitative Characteristics

Table 2 shows the total amount of annotated sentences along their six NE-categories and compares this with the three major public datasets for German NER. For our BIOfid dataset, we can see the high value of `TIME` and `TAXON` entities which, so far, do not exist for any publicly available dataset.

4.4.2 Data Format

We use the 4-column CoNLL-format which writes each word of a sentence horizontally along its lemma, POS tag and gold label, separating each sentence by an empty new line. For the tagging scheme, we opt for BIO (IOB2). Listing 1 shows an excerpt of the train file in which the entities `TIME`, `PERSON`, `LOCATION`, `TAXON` are marked by our team of annotators for a given sentence from the BIOfid corpus.

Listing 1: Sample sentence from BIOfid dataset

Mein	mein	PPOSAT	O
Sohn	Sohn	NN	O
konnte	können	VMFIN	O
am	an	APPRART	O
3	3	CARD	B–TME
.	––	$.	I–TME
1	1	CARD	I–TME
.	––	$.	I–TME
23	23	CARD	I–TME
den	der	ART	O
Fabrikanten	Fabrikant	NN	O
Walter	Walter	NE	B–PER
Schmidt	Schmidt	NE	I–PER
aus	aus	APPR	O
Geithain	Geithain	NE	B–LOC
bei	bei	APPR	O
einem	ein	ART	O
Spaziergang	Spaziergang	NN	O
auf	auf	APPR	O
dem	der	ART	O
Rochlitzer	Rochlitzer	NN	B–LOC
Berge	Berg	NN	I–LOC
auf	auf	APPR	O
eine	ein	ART	O
Ringamsel	Ringamsel	NN	B–TAX
,	––	$,	O
Turdus	Turdus	NN	B–TAX
torquatus	torquatus	ADJD	I–TAX
L	L	NN	B–PER
.	––	$.	O
,	––	$,	O
hinweisen	hinweisen	VVINF	O
.	$.	––	O

We split the BIOfid dataset into train, dev, test files by the common ratio of 80:10:10 percentages after randomizing its order of sentences. These final data files are utilized for training and evaluating our models, which are described in the next section.

4.5 Methods

For the evaluation of the BIOfid dataset, we use six different approaches and compare each others results: one classic *rule-based* model and five high-performing *embedding-based* models.

4.5.1 N-Gram Tagger for TR

We develop a naive sequence tagger as a baseline for the recognition of taxonomic entities in the BIOfid dataset. The baseline is only for a subtask of the full task of biological NER, described in the previous Section 4.1. Our sequence tagger is built on the k-skip-n-grams (with $k = 1$) which are constructed from the tokens of taxonomic entries in the comprehensive *Latin* and *German* gazetteers of biology. Both gazetteers consist of 83,348 taxonomic entries from various biological systematics such as of *aves*, *lepidoptera* and *vascular plant*. In addition, we consider *WikiData*[7] and construct an additional gazetteer by extracting 2,663,995 German and Latin taxonomic entries from the online resource by selecting all entries from a XML-dump that are subjects (`?s`) in the following two SPARQL triple patterns[8]:

- `?s instance-of taxon.`
- `?o subclass-of taxon.`
  `?s instance-of ?o.`

For each gazetteer entry consisting of at least three tokens ($n \geq 3$), we take all tokens as an input and create a list of 1-skip-n-grams. For example, for the taxonomic entry *iris kashmiriana b.*, we create four n-grams *(iris kashmiriana), (iris b.), (kashmiriana b.)* and *(iris kashmiriana b.)*. In this way, we construct 3,023,270 unique n-grams in total from 2,682,959 merged taxonomic entries, while dropping 140,432 duplicate n-grams entirely. Next, we map all these n-grams to the BIOfid test file by standard string matching and thus find the taxonomic occurrences in the target set of text data.

4.5.2 Neural Models for NER

Our neural models consist of two separately trained components: a) foundational word embeddings, modeling the general knowledge from large unlabeled text corpora, and b) various task-specific neural architectures, modeling the domain knowledge from the labeled training data. In this section, both components are presented briefly.

Word Embeddings The language model of continuous space word representations (*word2vec*) (Mikolov et al., 2013) and its variations by (Levy and Goldberg, 2014; Komninos and Manandhar, 2016) are the foundations of most ongoing research in NLP with neural networks. Based on the context, the model embeds words, phrases or sentences into high dimensional vector spaces. We use the model of *Wang2vec* (Ling et al., 2015) and its morphological extension (Ahmed and Mehler, 2018) which explores syntactic data specific for German and, thus, better suites the task of NER. We use the recently published German language word embeddings from the TTLab[9] which are pre-trained with the morphological extension of the Wang2vec algorithm on the COW corpus (Schäfer, 2015), the largest collection of German texts extracted from web documents with over 617 Mio. sentences. Out of the six published variants of embeddings, we opt for token-based embeddings (*COW.lower.wang2vec*), as they delivered the best results for German NER according to the publishers.

BiLSTM We provide a brief overview of the configurations for the five neural models which we use throughout this paper. The model *BiLSTM-CRF* is similar to the one used in (Ahmed and Mehler, 2018), which goes back to the work of (Lample et al., 2016). The neural network consists of stacked LSTM and CRF layers. The *base layer* combines for a given word its (pre-trained) word embedding with its character-based embedding. These features are forwarded to the *prediction layer* which produces the final NE tag.

Model	Emb.	Language Model	Train Data
BiLSTM-a	COW	N/A	BIOfid
Flair Wang2v.	COW	PCE	BIOfid
Flair ELMo	COW	PCE+Leipzig	BIOfid
Flair BERT	COW	PCE+BERT-Base	BIOfid
BiLSTM-b	COW	N/A	All

Table 3: Overview of the model inputs. For BiLSTM-b we consider all merged training data (i.e. BIOfid + GermEval + CoNLL)

Flair Wang2vec We further train a sequence labeling model using Flair[10]. We build the model in

[7] http://www.wikidata.org/

[8] All results can be acquired with the following WikiData queries: http://w.wiki/3u3 and http://w.wiki/3ud

[9] http://www.texttechnologylab.org/resources2018/

[10] http://github.com/zalandoresearch/flair

the same fashion as used by (Akbik et al., 2018) following the guide given by the authors for the task "CoNLL-03 Named Entity Recognition (German)", while keeping the pooled contextualized embeddings (PCE) and exchanging the GloVe embeddings employed by the authors with Wang2vec embeddings trained on the COW corpus.

Flair ELMo In addition to the previous model, we train a Flair Sequence Tagging model by stacking an ELMo embedding layer on top of the Flair Wang2vec model. The ELMo embeddings were trained on a section of the *Leipzig Corpora Collection* (Goldhahn et al., 2012) containing 100,000 sentences from Wikipedia using default parameters.

Flair BERT Similarly, we added BERT (Devlin et al., 2018) to the Flair Wang2vec model. We used the recently published *BERT-Base, Multilingual Cased*[11] pre-trained model for this purpose.

Hyperparameters We take the original neural models and keep the hyperparameters as described in their references. The only adjustments we make to the models are on the input level, i.e. we perform variations for the pre-trained word embeddings, the pre-trained language models, and the training data (see Table 3).

5 Results

We evaluate the performance of all models with the official script from the shared task of CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003). All our experiments were run on Nvidia's *GTX 1080 Ti* GPUs.

5.1 Baseline for TR

N-Gram Tagger Applying the gazetteer to the BIOfid test file gives us the respective baseline for the recognition of taxonomic entities. For evaluation, we use the CoNLL-script and contrast it with easing the conditions by evaluating only the NE predictions and ignoring the prefixed BIO-tagging scheme to every NE. The evaluation does not take into account the other words and is based only on the actual words annotated as `TAXON`.

Table 4 displays the results for the n-gram tagger. We can nicely see that the increase in size of gazetteers leads to an increase in the final performance. More specifically, for the eased condition, every incremental step from *Latin* to *Latin+German*, and the next step to *All* (i.e. *Latin+German+WikiData*) leads to an increase of $+6.64\%$ and $+6.56\%$ F-scores, respectively. This matter of fact demonstrates that for the n-gram tagger the resource-size matters.

Gazetteer	CoNLL-Eval	Pr. [%]	Re. [%]	F1 [%]
Lat.	standard	61.50	34.71	44.37
Lat.+Ger.	standard	65.83	45.42	53.75
WikiData	standard	69.05	53.91	60.55
Lat.	eased	92.48	46.04	61.06
Lat.+Ger.	eased	92.94	54.55	67.70
WikiData	eased	95.55	58.87	72.85
All	standard	69.20	55.75	61.75
All	eased	95.57	60.72	74.26

Table 4: Baseline for TR on the BIOfid test file with the N-Gram sequence tagger.

Furthermore, for the eased condition, we see very high scores for precision, however, the recall values are relatively low. This result demonstrates a classic problem of rule-based approaches; as there is no learning process involved, we assume that the performance of the n-gram tagger is highly limited on the features extracted from the source of knowledge (i.e. the amount of information contained in the gazetteer). Besides, no transfer learning is possible from related resources, demonstrating the downsides of non-learning methods.

5.2 Biological NER

We report here the results of our comprehensive survey of five current embedding-based high-performers for biological NER in historical biodiversity literature[12].

The Gold Standard Table 5 contains a detailed summary of all results. In that table, we the report the results which are given by T.H. Nguyen et al. (2019). For the optimized *BiLSTM Tagger*, we achieve excellent results and establish a new state-of-the-art for the first task of TR with 80.23% F-score (see Table 5: BiLSTM-a). For biological NER, we outperform the English counterpart *Co-*

[11] http://github.com/google-research/bert

[12] Our manual inspection of the training data showed that the annotations are content-wise homogeneous, except for the category `OTHER`. The annotators reported its usage as a residual NE-category for everything which is biologically interesting (e.g. morphology, animal behavior, reproduction, development) but does not fall under the definition of the five major categories. Initial experimental results confirmed its heterogeneous quality. Therefore we omitted `OTHER` (3,143 sentences) from our further experiments which in turn increased the final performance of NER.

Model	Scores [%]	TAXON	PERSON	LOCATION	ORGANIZATION	TIME	*Overall*
Copious Nguyen (2019)	Precision	77.42	58.92	85.05	N/A	70.67	77.49
	Recall	69.67	48.44	85.63	N/A	54.36	71.89
	F1	73.34	53.17	85.34	N/A	61.45	74.58
BiLSTM-a	Precision	81.33	63.19	66.20	60.24	91.16	75.62
	Recall	79.16	77.45	57.35	67.57	88.16	74.98
	F1	**80.23**	69.60	61.46	63.69	89.63	75.30
Flair Wang2vec	Precision	75.94	61.25	67.58	61.64	90.59	73.58
	Recall	81.37	76.09	62.89	58.11	85.24	75.89
	F1	78.08	71.89	62.63	56.95	87.89	74.30
Flair ELMo	Precision	75.64	67.16	58.31	56.82	90.49	73.05
	Recall	79.92	79.89	65.06	60.81	86.02	76.50
	F1	77.88	69.34	66.30	61.22	88.25	75.01
Flair BERT	Precision	76.63	65.30	66.96	58.00	92.21	74.98
	Recall	77.38	81.02	61.89	58.00	90.33	76.22
	F1	77.01	72.31	64.32	58.00	**91.26**	75.59
BiLSTM-b	Precision	80.45	88.61	72.72	81.21	87.63	79.35
	Recall	76.65	89.40	84.02	70.74	81.17	75.38
	F1	78.50	**89.00**	**77.96**	**75.61**	84.27	**77.31**

Table 5: Results for the task of German biological NER with various neural networks models along the English baseline on the Copious dataset (T.H. Nguyen et al., 2019). All models are trained on the BIOfid dataset and evaluated with the official CoNLL-2003 eval script.

pious for all categories except for LOCATION. For the latter category, the *Copious* dataset contains *9,921* training samples whereas ours has 3,136 fewer samples. We assume that this lower amount results into the lower performance.

With the popular deep language models *Flair*, *ELMo* and *BERT*, we interestingly stay below the performance of the BiLSTM model (except for TIME). Although we utilize the same pre-trained COW word embeddings for all models, we assume that the lower performance arises due to the language models themselves being trained on only a relatively small corpus (ELMo: 100,000 sentences). However, for training ELMo on larger corpora, such as the COW corpus, we would require many months of training time. For the pre-trained Flair and BERT, we can only fine-tune the last tagging layer, not the whole language model itself. This stands in contrast to the BiLSTM model which can be wholly targeted to our domain-specific training data. Hence, this demonstrates the downside of such heavy language models; although they might deliver the top performances, it is difficult to adjust them for lightweight processes, making them impractical for the context of low-resources scenarios.

Data Merging for BiLSTM Tagger For BiLSTM-a, it can be noted that the performance of the standard categories PERSON, ORGANIZATION, and, especially LOCATION is inferior. Therefore, we performed resource-optimization by merging high quality data with our BIOfid dataset in order to increase the training samples for the low performing categories. We merge the datasets of GermEval and CoNLL with our annotated sentences, resulting in train, dev, and test sizes of *46,857*, *6,629*, and *9,437* sentences, respectively. Table 5: BiLSTM-b shows the improvements in performance with the increased dataset. Our results demonstrate the effectiveness of our approach; we do not need to modify the model, rather it is sufficient to perform data-driven optimization. Considering the overall performance, we outperform the English counterpart by $+2,73\%$ F-Score and thus establish a new state-of-the-art for the task of biological NER.

Error Analysis We manually analyze the errors made by the ensemble of neural models. We observe three major issues that compose the absolute majority of errors: a number of *missing annotations* from our experts, *OCR erros* in the raw text and *rare words* that occur frequently in our test dataset. An example of an OCR error is the annotated text span *[1, Juni 1967]* TIME which is misclassified by all models as *1, [Juni 1967]* TIME due to the comma in the date format. Another example is *[KLeebend]* LOC which is not tagged due to the capital "L". Further, the word *[Venn-*

fußfläche] LOC occurs 17 times in the test dataset, but only twice in the training set. It is a three word compound of the words *Venn*, *Fuß* and *Fläche*, that describes a part of the landscape *Vennvorland* in Germany. We conclude that the preprocessing pipeline has to be further refined to remove the OCR errors, while a re-annotation of the data could solve the missing annotations and a more thorough shuffle may solve the rare word issue.

6 Conclusion

In this study, we presented a newly annotated *BIOfid* dataset for German NER in historical biodiversity literature and performed a comprehensive evaluation of the quality of our dataset with five competing neural models. We come to the conclusion that the value of our dataset does not rely solely on the two new entities of TIME and TAXON. By generating domain-related annotation data typical for historical biodiversity literature, we increase the potential performance for biological NER, even for the four standard NE categories. This was demonstrated by the limited scope of the rule-based approach which could not come close to the performance delivered by the neural models and which, in turn, established a new state-of-the-art for both of our selected tasks of TR and NER.

In the course of the annotation process, we discovered that there are further information entities in the BIOfid corpus which do not fall into the definition of standard NE-categories, albeit they are useful from the perspective of biodiversity researchers. For future work, we plan to increase the semantic granularity of the BIOfid dataset by mapping and re-annotating the existing six NE-categories to the top-level hierarchy of *WordNet* (Miller, 1995). This includes 26 categories that can be either *abstract entities* or *concrete entities* (i.e. NE) and can be assigned to specific biological entities, such as morphology, habitat, reproduction, behavioral traits, or species community. By re-annotating the dataset we additionally plan to deliver an inter-agreement value for both the current NER-dataset and the much smaller WordNet-dataset (which is planned to contain an up to 9 times higher amount of annotated information per sentence). Furthermore, we plan to extract all biological entities with the trained neural models from the BIOfid corpus and perform on them the task of *relation extraction* based on current embedding methods.

Overall, our work mobilizes data from undigitized literature leading to huge potentials for biodiversity researchers. It enables cartographic research on the distribution of Central European biodiversity ranging from the pre-modern time up to our current ever increasingly digitizing age.

Acknowledgments

This work was funded by the German Research Foundations (DFG) as part of the *BIOfid* project[13] (DFG-326061700). The BIOfid dataset, the taxonomic gazetteers, the source code, and the supporting appendix (which includes the full annotation guidelines) are uploaded on GitHub[14] for the research community. Special thanks go to W. A. Hemati and G. Abrami for setting up the annotation environment, and to R. Roller, S. Löbbert, Dr. G. Kasperek, Dr. A. Hausinger, Dr. T. Hörnschemeyer and further project members for supporting the annotation process.

References

Giuseppe Abrami, Alexander Mehler, Andy Lücking, Elias Rieb, and Philipp Helfrich. 2019. TextAnnotator: A flexible framework for semantic annotations. In *Proceedings of the Fifteenth Joint ACL - ISO Workshop on Interoperable Semantic Annotation, (ISA-15)*, ISA-15.

Sajawel Ahmed and Alexander Mehler. 2018. Resource-Size matters: Improving Neural Named Entity Recognition with Optimized Large Corpora. In *Proceedings of the 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Darina Benikova, Christian Biemann, and Marc Reznicek. 2014. NoSta-D Named Entity Annotation for German: Guidelines and Dataset. In *LREC*.

Armin Burkhardt. 2004. 2004. Nomen est omen? : zur Semantik der Eigennamen. In *Landesheimatbund Sachsen-Anhalt e. V. (Hrsg.): "Magdeburger Namenlandschaft" : Orts- und Personennamen der Stadt und Region Magdeburg*.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named Entity Recognition: A Maximum Entropy Approach

[13] http://biofid.de/en/

[14] https://github.com/texttechnologylab/BIOfid

Using Global Information. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Dirk Goldhahn, Thomas Eckart, and Uwe Quasthoff. 2012. Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages. In *LREC*.

Alexandros Komninos and Suresh Manandhar. 2016. Dependency Based Embeddings for Sentence Classification Tasks. In *HLT-NAACL*, pages 1490–1500.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *NAACL-HLT*.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *ACL (2)*, pages 302–308.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/Too Simple Adaptations of word2vec for Syntax Problems. In *NAACL-HLT*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Roland Schäfer. 2015. Processing and querying large web corpora with the COW14 architecture. In *Proceedings of Challenges in the Management of Large Corpora 3 (CMLC-3)*, Lancaster. UCREL, IDS.

A Schiller, S Teufel, C Stöckert, and C Thielen. 1999. Guidelines für das Tagging deutscher Textcorpora mit STTS [Guidelines for tagging German corpora of written language with STTS]. Technical report, Technical Report. Stuttgart, Germany: Institut für maschinelle Sprachverarbeitung [Institute for Machine Language Processing].

Stefan Schweter and Sajawel Ahmed. 2019. Deep-EOS: General-Purpose Neural Networks for Sentence Boundary Detection. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS)*. Accepted.

Heike Telljohann, Erhard W Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck. 2012. Stylebook for the Tübingen treebank of written German (TüBa-D/Z).

Scott A Thomson, Richard L Pyle, Shane T Ahyong, Miguel Alonso-Zarazaga, Joe Ammirati, Juan Francisco Araya, John S Ascher, Tracy Lynn Audisio, Valter M Azevedo-Santos, Nicolas Bailly, et al. 2018. Taxonomy based on science is necessary for global conservation. *PLoS biology*, 16(3):e2005075.

Nhung T.H. Nguyen, Roselyn S. Gabud, and Sophia Ananiadou. 2019. COPIOUS: A gold standard corpus of named entities towards extracting species occurrence from biodiversity literature. *Biodiversity Data Journal*, 7:e29626.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Slang detection and identification

Zhengqi Pei[1,2] **Zhewei Sun**[3] **Yang Xu**[3]

[1]Engineering Science Program, University of Toronto
[2]HAETEK Institute of Machine Intelligence, Shenzhen, China
[3]Department of Computer Science, University of Toronto
zhengqi.pei@mail.utoronto.ca
{zheweisun, yangxu}@cs.toronto.edu

Abstract

The prevalence of informal language such as slang presents challenges for natural language systems, particularly in the automatic discovery of flexible word usages. Previous work has explored slang in terms of dictionary construction, sentiment analysis, word formation, and interpretation, but scarce research has attempted the basic problem of slang detection and identification. We examine the extent to which deep learning methods support automatic detection and identification of slang from natural sentences using a combination of bidirectional recurrent neural networks, conditional random field, and multilayer perceptron. We test these models based on a comprehensive set of linguistic features in sentence-level detection and token-level identification of slang. We found that a prominent feature of slang is the surprising use of words across syntactic categories or syntactic shift (e.g., verb→noun). Our best models detect the presence of slang at the sentence level with an F1-score of 0.80 and identify its exact position at the token level with an F1-Score of 0.50.

1 Introduction

Slang, or 'the language of streets' (Green, 2015), is a type of informal language consisting of words and expressions shared within specific groups. A hallmark of slang is its expressivity, instantiated in the flexible use of words. For example, the word *sick* with the conventional sense of "ill" can also denote a positive slang sense of "awesome", such as "the band's album is sick". The expressive nature of slang exemplifies its social function, because it provides an effective way of communicating and knowledge-sharing within groups of distinct social identities, such as in the cases of vulgar tongue (Green, 2015) and online language. On the other hand, the flexible nature of slang use can be intriguing for language users, learners, and natural language systems. Here, we ask whether slang can be automatically detected in natural sentences, and what linguistic features might distinguish slang usage from conventional language use.

Our problem statement is simple: Given a natural sentence such as "the band's album is sick", can machines learn to 1) detect whether slang usage is present or not (i.e., sentence-level detection), and 2) identify the exact position of the slang term in the sentence (i.e., token-level identification). For each of these tasks, our systems should be able to learn to cope with two main categories of slang usage (Dhuliawala et al., 2016):

- **Newly extended senses**: existing words in the lexicon that develop novel slang senses distinct from their conventional senses, e.g., *clutch* refers to "an act of grasping" in its conventional usage, but is later extended to the slang sense of "tense critical situation".

- **Newly created words**: words that do not exist in the standard lexicon, e.g., blending of *friend* and *enemy* forms the slang word *frenemy* that describes a person who is simultaneously friend of and in conflict with someone.

Research on slang in the natural language processing community falls under several categories, but to our knowledge the current work is the first to tackle the basic problem of automatic slang detection and identification.

2 Related computational work

2.1 Slang dictionary construction and sentiment analysis

Existing approaches such as SlangNet (Dhuliawala et al., 2016), SlangSD (Wu et al., 2018), and SLANGZY (Gupta et al., 2019) have focused on efficiently maintaining and extending the construction of slang dictionaries to aid computational

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 881–889
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

sentiment analysis of slang content. Some popular systems from this line of research are based on modular representation (Pal and Saha, 2013) that treats slang in terms of various linguistic stages, each of which deals with slang word from different aspects, e.g., sound, concept, formation, etc. These dictionary-based methods rely on static lexical information and structure, which are typically not sufficient to capture the flexible semantics and lexical coinage in natural slang usages.

2.2 Slang word formation and interpretation

An independent line of work has explored generative models (Kulkarni and Wang, 2018) for slang word formation that captures processes such as blending, clipping, and reduplication. This work uses long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) models to generate slang words in terms of string components according to type-sensitive characteristics. Related work has also explored automatic interpretation of non-standard English words and phrases using sequence-to-sequence architecture with dual encoders (Ni and Wang, 2017). This method generates literal interpretations for queried non-standard expressions from source sentences, but the primary focus is on explanation as opposed to detection or identification, both of which are prerequisite tasks for slang interpretation.

Differing from these existing research, we present a methodological framework based on standard techniques in deep learning for automatic slang detection and identification that does not rely heavily on dictionary construction. We examine a comprehensive set of linguistic features that might be diagnostic of slang usage in natural settings, and we explore existing methods that leverage bidirectional LSTM with multilayer perceptron (MLP) (Rauber and Berns, 2011) and conditional random field (CRF) (Lafferty et al., 2001). Our framework is related to existing work that applies sequence-to-sequence models with attention mechanism (Luong et al., 2015) for the identification of dialectal varieties (Jurgens et al., 2017) and feature-based emotion detection from online media (Ileri and Karagoz, 2016). However, our emphasis here is on learning features that are relevant to the automatic discovery of slang usage.

To preview the framework, our models formulate slang detection and identification as a sequence-labelling task. In addition to typical word embedding inputs, we incorporate relevant linguistic features in the input via an efficient feature boosting procedure. Throughout our experiments, we found that the flexibility of Part-of-Speech (POS) feature is most diagnostic of slang usage: Slang often entails structured POS transformation of existing syntactic uses of words. We show how features related to POS confidence and POS shift in the input provide the improvement on model performance. We also demonstrate how novel tokens of slang can be discovered using a character level convolutional neural network (Zhang et al., 2015).

3 Computational methodology

We present the models and features we use for machine detection and identification of slang.

3.1 Specification of predictive tasks

In the *slang detection* task, our models determine whether a given sentence contains at least one slang usage, which can be an existing word with a novel slang meaning or a newly created word. We formulate this as a binary classification task.

In the *slang identification* task, our models identify each token within the input sentence as 'non-slang' or 'slang' by sequence labeling, which determines the exact positions of slang usage. Note that the models in the identification task encapsulate the detection task; an empty prediction that labels all tokens as 'non-slang' is equivalent to classifying the sentence as a non-slang sentence in the detection task, and vice versa.

3.2 Model architectures

We present a BiLSTM-MLP model that is capable of identifying slang words in a given sentence. The basic architecture is shown in Figure 1. Fully connected MLP layers are placed on top of both the forward and backward hidden states H_f and H_b of a biLSTM network encoding the input sentence. The output of the two MLP layers f_L and b_L are then concatenated as an input to the final MLP layer. In total, there are three components $W^{(f)}_{MLP}$, $W^{(b)}_{MLP}$, $W^{(con)}_{MLP}$ within the MLP block that are shared across all hidden states:

$$f_L = \sigma(W^{(f)}_{MLP} H_f + b^{(f)}) \quad (1)$$

$$b_L = \sigma(W^{(b)}_{MLP} H_b + b^{(b)}) \quad (2)$$

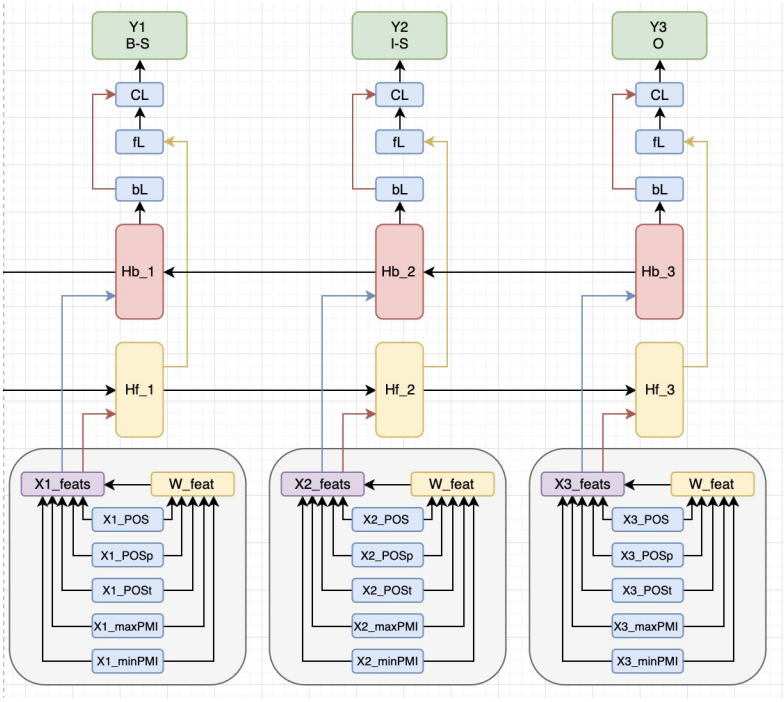


Figure 1: **BiLSTM-MLP model architecture with feature boosting.** The architecture for bidirectional LSTM with Multilayer Perceptron (MLP) as the output layer. The shared bL layer is the MLP unit for the current backward hidden state; the shared fL layer is the MLP unit for forward hidden state; the shared CL layer takes as input the concatenation of the outputs from fL and bL, and output the predictive tagging distribution.

$$C_L = \sigma(W_{MLP}^{(con)}[f_L; b_L] + b^{(con)}) \quad (3)$$

$$Y = softmax(W^{(Y)}C_L + b^{(Y)}) \quad (4)$$

The resulting output vectors are subsequently used to compute the predictive tag for input tokens. There are seven tags here: 'START', 'END', 'O', 'B-U', 'I-U', 'B-N', 'I-N'. These tags apply 'BIO' convention (Ramshaw and Marcus, 1999) that labels non-target token as 'O', initial token of the interested region (e.g., phrase) as 'B-', and the subsequent intermediate tokens of interest as 'I-', etc.

The MLP block in BiLSTM-MLP model can be swapped with an alternative conditional random field (CRF) (Lafferty et al., 2001) that better considers explicit sequential restrictions, e.g., the tag 'I-U' has to be placed after a 'B-U' tag. This sequential restriction can be captured via combination of an LSTM network and a CRF network. The CRF layer has a state transition matrix $A_{(i,j)}$ that models the transition score between the i-th tag and the j-th tag, and an emission matrix $f_{(i,k)}$ that models the output score for the i-th tag at the k-th word (Huang et al., 2015). The source sentence $X_{(i)}$ along with a sequence of tags $Y_{(i)}$ is evaluated via a CRF score:

$$S(X, Y) = \sum_{t=1}^{T}(A_{Y_{t-1},Y_t} + f_{Y_t,X_t}) \quad (5)$$

The CRF layer uses output states from BiLSTM layer to find tags in sequence with optimal CRF score to make prediction. (i.e. $X = [H_f; H_b]$) The CRF probability is easily computed in favor of the logarithmic predictive score as follows:

$$log(P(Y|X)) = S(X, Y) - log(\sum_{Y' \in Y_x} exp(S(X, Y'))) \quad (6)$$

Analogous to Huang et al. (2015), we apply dynamic programming during training to handle the intractable summation term. This BiLSTM-CRF model aims to identify the exact position of each slang word, in terms of sequential restrictions.

3.3 Linguistic features

We use a comprehensive set of linguistic features to facilitate interpretable learning from the models described. Carefully curated linguistic features can improve the training efficiency because linguistic knowledge helps to rectify the learning

process. Feature-based inputs support mapping between contextual concepts and domain-specific clues via distributed representations. The following linguistic features are stored with unique entries in the lookup tables, and they are encoded into embeddings via distributed representation. Figure 2 illustrates these features for the example token *fire*.

Unigram. We take each individual word as an input to the models. The words are represented by standard multi-dimensional word vectors obtained via embedding models such as word2vec (Mikolov et al., 2013).

Bigram. Similar to unigram embedding, a bigram embedding represents the word vector for a bigram. For instance, given an arbitrary word W_t at time-step t in its source sentence, the bigrams for W_t are $W_{t-1}W_t$ and W_tW_{t+1}, which correspond to forward bigram and backward bigram respectively. Whereas the unigram embedding X_t is identified as the word vector representation for W_t, the bigram embeddings are defined as their concatenations. The forward bigram embedding fB_t represents the vector $[W_{t-1};W_t]$, and the backward bigram embedding bB_t represents the vector $[W_t;W_{t+1}]$. Note that both forward and backward bigrams are implemented using the identical lookup table.

Pointwise Mutual Information. We consider measurement of discrepancy between two linguistic variables via PMI (Aji and Kaimal, 2012). Given two source words W_i and W_j, the PMI between them is computed as follows:

$$PMI(W_i, W_j) = log\frac{Pr(W_i, W_j)}{Pr(W_i)Pr(W_j)} \quad (7)$$

We estimate the probabilistic distributions from the Penn Treebank (Marcus et al., 1993), where the probabilities of the PMI can be computed based on co-occurrence statistics. In our models, we compute PMI of the current word with each of its neighboring words, and encode the resulting maximum and minimum PMIs as the features. For example, given $PMI(W_i, W_{i-1}) = 0$ and $PMI(W_i, W_{i+1}) = 2$, we would have the PMI features related to the current word W_i as $maxPMI = 2$ and $minPMI = 0$.

Part-of-Speech. We consider Part-of-Speech (POS) that represents a word's syntactic category. Common POSs include noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection, and numeral. Multiple POSs can be assigned to an identical word due to the possibility of a word having distinct grammatical properties in different sentences, e.g., *work* is a verb in "these models work", but it is a noun in "I don't like this work". We use Natural Language Toolkit (NLTK) (Loper and Bird, 2002) for POS tagging.

POSp. This linguistic feature is an accessory feature to the POS feature that only represents the grammatical property for the token in its current semantic context. Since each word token might have multiple POS tags, it is possible to count a word's POS distribution that represents the probabilities of a word attached with this specific POS tags as an additional linguistic feature. For example, given a well-formed text corpus C:

$$Pr(POS(like) \leftarrow verb|C) = 0.8$$
$$Pr(POS(like) \leftarrow noun|C) = 0.1$$
$$Pr(POS(like) \leftarrow numeral|C) = 0.0$$

POSt. Word class transfer is a common mechanism (e.g., in English) for extending word senses. We consider a novel feature that represents the transformation from the root-POS (the most commonly used POS for the current token) to the current-POS for the token, e.g., "IN-VB" is a POSt feature, where "IN" is the root-POS, "VB" is the current-POS of the token.

Bigram-Count. The Bigram-Count is similar to the $POSp$, except that the Bigram-Count represents the probability that the current word is collocated with its neighboring words. Given an arbitrary word token W_i in a sequence ϕ, the forward and backward Bigram-Counts are evaluated as

$$fBC = log(\frac{Pr(W_{i-1}W_i)}{Pr(W_i)}) \quad (8)$$

$$bBC = log(\frac{Pr(W_iW_{i+1})}{Pr(W_i)}) \quad (9)$$

The Bigram-Counts are also similar to PMI except that the Bigram-Counts focus more on the current word. In some cases, the Bigram-Count will leverage the zero PMI of low-frequent collocations.

3.4 Feature boosting: Feature-level learning

We present feature boosting for the models with limited features to learn feature-level knowledge. The linguistic features are assumed to be related in terms of the concatenated form of input vectors. For an input token, the related features can

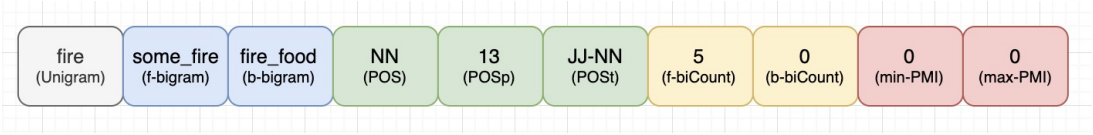


Figure 2: **Concatenation of linguistic features for a given token.** For a specific token *fire* in the source sentence "she can cook some fire food", the related linguistic features are represented as token vectors to concatenate the feature-based input for this token. Each randomly initialized vector is updated during training. The unigram features are represented as 32-dimensional word vectors, the bigram vectors are 20-dimensional, and all else are 16-dimensional.

be assigned with distinct weights that selectively focus on specific features. Suppose we have a token x represented by k distinct linguistic features $[f_1^{(x)}...f_k^{(x)}]$, where $f_i^{(x)} \in \mathbb{R}^{|V_i|\times 1}$, and a shared multi-layer perceptron W_{feat} across all the states. The local feature weights are defined as:

$$\alpha_{feat}{}^{(x)} = W_{feat}[f_1^{(x)}...f_k^{(x)}] + b_f \qquad (10)$$

Where b_f is the MLP bias, and $\alpha_{feat}{}^{(x)}$ is k-dimensional vector that assigns distinct weights to each feature. To provide feature-level information for the vectors fed as inputs to the BiLSTM layer, the feature vectors are weighted in terms of the computed $\alpha_{feat}{}^{(x)}$ for concatenation:

$$V_x = [f_1^{(x)} \cdot \alpha_{feat_1}{}^{(x)}...f_k^{(x)} \cdot \alpha_{feat_k}{}^{(x)}] \qquad (11)$$

The concatenation of weighted feature embeddings contains global feature-level information, allowing the inputs to selectively feed into the model in terms of the feature distribution. As an alternative, the last propagated hidden states from both forward and backward layers of the BiLSTM can be concatenated with the raw features $f_i^{(x)}$ to compute the local feature weights.

We demonstrate later the relative importance of different features in a feature ablation analysis where we remove less relevant input features and keep only the light-weighted but informative linguistic features such as Part-of-Speech and POSp.

3.5 Treatment of novel slang word forms

Our models are able to handle novel tokens by learning the contextual structure within a sentence. Although all the out-of-vocabulary tokens are consistently labelled as 'UKT' such that they are truly unseen by the models, the sequential relations can be captured by the hidden layers of LSTMs. In order to improve model predictability on unknown tokens, we apply character-based convolutional neural network to encode the spelling of words.

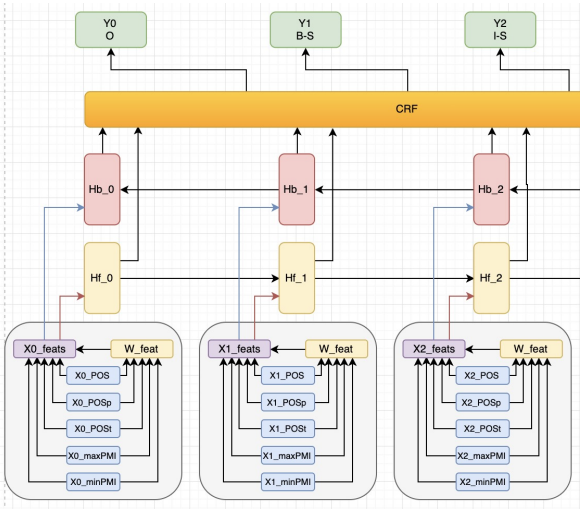


Figure 3: **BiLSTM-CRF model architecture with feature boosting.** The architecture for bidirectional LSTM with Conditional Random Field as the output layer. The shared CRF layer takes as input the BiLSTM's outputs from hidden states, then finds the optimal path in terms of sum of emission scores and transition scores. The optimal path results in the prediction.

Each character is represented by a fixed dimensional embedding (Zhang et al., 2015), similar to word embeddings, and forwarded into a convolutional neural network to obtain a character-level encoding of the word. The resulting Char-CNN embeddings are concatenated with the original input embeddings that feed into the models.

4 Experiments and results

4.1 Experimental setup

We consider datasets that are composed of sentences in two distinct categories, standard (slang-less) and slang-specific:

- **Slang-less sentence dataset**: 15-thousand non-slang sentences from Wall Street News (2011-2016) in Penn Treebank (Marcus et al., 1993) as the negative examples.

- **Slang-specific sentence dataset**: 15-thousand sentences that contain slang words from Online Slang Dictionary (http://onlineslangdictionary.com/) as the positive examples.

The sentences from Wall Street News are taken to be non-slang sentences since the news-based sentences were typically standard English conformed and reviewed before publication. In order to construct an even more trustworthy negative set for standard English, we filtered the sentences from Wall Street News based on the proportion of unknown tokens within the sentences. A News sentence will be considered as an eligible non-slang sentence if it has at most 20% words that have not been covered by the provided frequency-based vocabulary. The vocabulary (or lexicon) consists of top 25,000 most frequent English words from an authoritative text corpus, e.g., Penn Treebank. On average, each negative example sentence contains 12.11 (mean) tokens with standard deviation of 2.52.

We collect positive examples from lexical entries in the Online Slang Dictionary (OSD) where example usage sentences are available. We obtain the ground-truth slang usage positions from OSD and apply the BIO tagging scheme, which labels interested tokens with "B-" at the beginning token, and with "I-" at the subsequent tokens. All the tokens out of interest are labelled as "O". There are two kinds of slang types: UKT-slang and Normal word slang, labelled with "U" and "N", respectively. The UKT-slang refers to slang usages with novel word forms, while normal word slang refers to slang usages with existing words. Out of the 15,000 positive examples, 10,000 of which contain UKT slang words that are not covered by the lexicon. On average, each positive example sentence contains 13.79 (mean) tokens with standard deviation of 3.42.

All models are trained using the Adam optimizer (Kingma and Ba, 2015) with a learning rate 0.001 with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

4.2 Evaluation and results

We evaluate our models in terms of slang detection and identification. We also perform feature ablation to locate salient features of slang usage and show example cases of model success and failure.

4.2.1 Detection task

We evaluated our models to determine whether a given sentence contains at least one slang usage. The evaluation metrics are precision, recall, and F1-score. Table 1 summarizes the results from the models. Overall, all our proposed models performed substantially better than the baseline, with the CRF-based models yielding better performance than the MLP-based models. For instance, although all models tend to have high precision and relatively lower recall, the CRF-based models generally achieve better recall than the MLP-based models given the same level of precision. Importantly, the best overall model makes use of all linguistic features and yields an F1-score close to 0.80. This result suggests that the features we proposed contribute critically to both the precision and recall of slang detection. It is worth noting that models with only the POS related features achieve reasonable performance (although not as well as the full model), and we will return to this observation in the ablation analysis.

4.2.2 Identification task

We evaluated our models to identify each word in a given sentence at the word level. The evaluation metrics are precision, recall, and F1-score. Table 2 summarizes the results. Similar to the case of detection, our models performed substantially above the baseline in this task. In particular, both the MLP and CRF-based models yielded higher F1 scores (close to 0.50) when multiple features are taken into account. Tables 3 and 4 further summarize the results (i.e., number of correctly predicted cases) of these models in identifying the two main different types of slang: novel slang word and novel slang sense (of an existing word), and how the models fair with and without the incorporation of CNN character-based embeddings.

4.2.3 Feature ablation

We evaluated the contributions of the linguistic features on the test set via model performance degradation through ablation. We would like to evaluate the extent that the model performance would be degraded with respect to a single feature (e.g., POS), given a trained model with the complete featured set. In this case we would force all the POS embeddings to be zero-vectors, and we then compare the ablated model against the full model. Figure 4 summarizes the degradation of

Model (features)	Precision	Recall	F1-score
Random Guess (baseline)	0.5000	0.5000	0.5000
BiLSTM-MLP (POS+POSp)	**0.9893**	0.4806	0.6469
BiLSTM-MLP (POS+POSp+POSt+PMI)	0.9777	0.5651	0.7162
BiLSTM-MLP (POS+POSp+POSt+PMI boosting)	0.9771	0.6053	0.7475
BiLSTM-MLP (full features)	0.9433	0.6842	0.7931
BiLSTM-CRF (POS+POSp)	0.9873	0.5969	0.7440
BiLSTM-CRF (POS+POSp+POSt+PMI)	0.9749	0.6482	0.7787
BiLSTM-CRF (POS+POSp+POSt+PMI boosting)	0.9749	**0.6496**	**0.7797**
BiLSTM-CRF (full features)	0.9518	**0.6856**	**0.7971**

Table 1: Model comparisons in the slang detection task.

Model (features)	Precision	Recall	F1-score
Random Guess (baseline)	0.0263	0.4834	0.0498
BiLSTM-MLP (POS+POSp)	**0.6240**	0.3172	0.4206
BiLSTM-MLP (POS+POSp+POSt+PMI)	0.6172	0.3864	0.4753
BiLSTM-MLP (POS+POSp+POSt+PMI boosting)	0.5967	0.3975	0.4771
BiLSTM-MLP (full features)	0.5423	**0.4612**	**0.4985**
BiLSTM-CRF (POS+POSp)	0.5666	0.3712	0.4485
BiLSTM-CRF (POS+POSp+POSt+PMI)	0.5763	0.4183	0.4847
BiLSTM-CRF (POS+POSp+POSt+PMI boosting)	0.5954	**0.4280**	**0.4980**
BiLSTM-CRF (full features)	0.5499	0.4501	0.4950

Table 2: Model comparisons in the slang identification task.

Model (features)	New Word	New Sense
BiLSTM-MLP (POS+POSp)	194/523	35/199
BiLSTM-MLP (POS+POSp+POSt+PMI)	228/523	53/199
BiLSTM-MLP (POS+POSp+POSt+PMI boosting)	236/523	**53/199**
BiLSTM-MLP (full features)	**267/523**	66/199
BiLSTM-CRF (POS+POSp)	227/523	41/199
BiLSTM-CRF (POS+POSp+POSt+PMI)	251/523	50/199
BiLSTM-CRF (POS+POSp+POSt+PMI boosting)	**260/523**	50/199
BiLSTM-CRF (full features)	242/523	**83/199**

Table 3: Model comparisons on identified slang by type (either as new word or existing word with new sense).

Model	Identification F1-score	Detection F1-score
BiLSTM-MLP	0.5101	0.8649
BiLSTM-CRF	0.5024	0.8679
BiLSTM-MLP with Char-CNN	**0.5172**	**0.8693**
BiLSTM-CRF with Char-CNN	0.5146	0.8679

Table 4: Comparisons of model performance with and without character-based embedding.

model performance based on ablation of each individual feature in question.

Salience of Part-of-Speech transformation. Based on the feature ablation analysis, Part-of-Speech features are the most crucial to overall model performance. Bigram counts come next which suggests that syntacmatic relations also play a role in slang usage. We probed the most prominent features by dividing the POS transformations observed in the data into two kinds: homogeneous and heterogeneous transformations. POSt features such as "VV-VV" (i.e. the identified

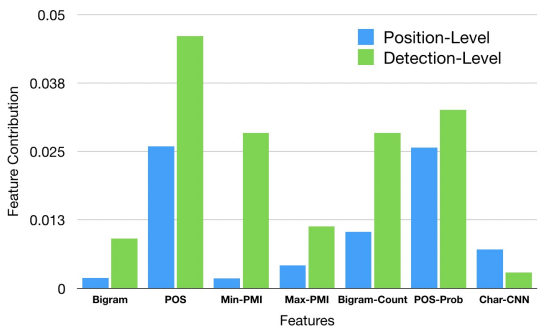


Figure 4: **Summary of feature ablation analysis.** The contributions of the individual linguistic features are shown. The degradation in performance is considered to be equivalent to a feature's contribution.

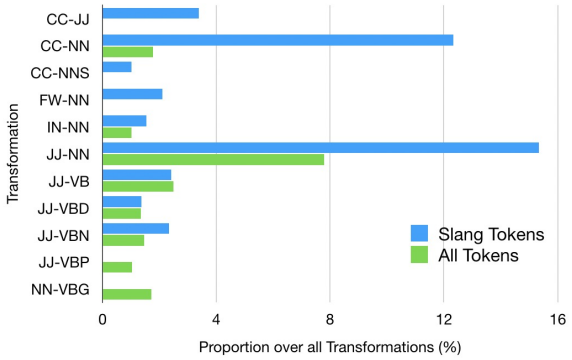


Figure 5: **Transformation of Part-of-Speech in slang usage.** Heterogeneous POS transformations (POSt) that have proportions higher than 1% are shown. There are 11 distinct POSt, 9 of which correspond to cases where the POSt proportion of slang word usage is higher than that of common word usage (i.e., control set). Both the slang tokens and normal tokens with JJ (adjective) tend to transfer to NN (noun); the slang words with CC (coordinating conjunction) are more likely to tranfer to NN (noun).

POS given the source sentence is identical to the root POS) are considered homogeneous transformations; the heterogeneous POSt refer to the case when the target POS differs from the root POS. The proportion of heterogeneous POSt over all the transformations is 25.74% among all the tokens, while the heterogeneous proportion surprisingly increases to 53.94% in slang-specific tokens. This indicates that a slang word is twice as likely to experience (heterogeneous) POS transformation in comparison to an arbitrary word, providing evidence that syntactic shift is a salient feature of slang usage. A comparison between POSt distributions of slang-specific and ordinary use cases is shown in Figure 5.

4.2.4 Examples of model success and failure

We provide examples of both successful and failed predictions to demonstrate the model capability in slang detection and identification:

- **Probe sentence**: "That money you sent me was clutch."
- **Model prediction**: ["clutch"]
- **Ground truth**: ["clutch"]

In the probe sentence, the token *clutch* refers to tense critical situation (noun) rather than its common sense "grasping" (verb). Our model successfully detected this slang component in the query.

- **Probe sentence**: "That's a real blower."
- **Model prediction**: ["–"] (no slang detected)
- **Ground truth**: ["blower"]

The token *blower* normally refers to a device that produces a current of air in common usage, while it refers to "surprise" in this probe sentence. Our model failed to detect this slang-component possibly due to insufficiency of contextual information.

5 Discussion

We take an initial step at automatic detection and identification of slang from natural sentences using established deep learning methods. We show how linguistic features combined with deep learning algorithms offer interpretability. We find that the bidirectional LSTM with feature-based inputs and character-based convolutional embeddings using multilayer perceptron yield the best performance in position identification, and the model with similar mechanisms except with conditional random field has better performance in detecting whether a source sentence contains a slang term. For unknown tokens, character-based convolutional embeddings improve the model in handling novel slang terms. We demonstrate that features combined with distributed word embeddings help machine detection of slang in general, and that Part-of-Speech among others is a prominent feature of slang usage. Our work provides a basis for locating slang from its flexible and unconventional syntactic word uses and offers opportunities for slang processing in downstream tasks in natural language processing.

References

Subhanpurno Aji and Ramachandra Kaimal. 2012. Document summarization using positive pointwise mutual information. *International Journal of Computer Science Information Technology*, 4:47–55.

Shehzaad Dhuliawala, Diptesh Kanojia, and Pushpak Bhattacharyya. 2016. SlangNet: A WordNet like resource for English slang. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 4329–4332, Portorož, Slovenia. European Language Resources Association (ELRA).

Jonathon Green. 2015. *The vulgar tongue: Green's history of slang*. Oxford University Press, USA.

Anshita Gupta, Sanya Bathla Taneja, Garima Malik, Sonakshi Vij, Devendra K. Tayal, and Amita Jain. 2019. Slangzy: a fuzzy logic-based algorithm for english slang meaning selection. *Progress in Artificial Intelligence*, 8(1):111–121.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *Arxiv Preprint*.

Ibrahim Ileri and Pinar Karagoz. 2016. Detecting user emotions in twitter through collective classification. *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*.

David Jurgens, Yulia Tsvetkov, and Dan Jurafsky. 2017. Incorporating dialectal variability for socially equitable language identification. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Vivek Kulkarni and William Yang Wang. 2018. Simple models for word formation in slang. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, LA, USA. ACL.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Ke Ni and William Yang Wang. 2017. Learning to explain non-standard english words and phrases. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*, Taipei, Taiwan. AFNLP and ACL.

Alok Ranjan Pal and Diganta Saha. 2013. Detection of slang words in e-data using semi-supervised learning. *International Journal of Artificial Intelligence and Applications*, 4(5):4961.

Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.

T. Rauber and K. Berns. 2011. Kernel multilayer perceptron. In *2011 24th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 337–343.

Liang Wu, Fred Morstatter, and Huan Liu. 2018. Slangsd: Building, expanding and using a sentiment dictionary of slang words for short-text sentiment classification. *Lang. Resour. Eval.*, 52(3):839–852.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.

Alleviating Sequence Information Loss with Data Overlapping and Prime Batch Sizes

Noémien Kocher[♢] Christian Scuito[♠] Lorenzo Tarantino[♠] Alexandros Lazaridis[♡]
Andreas Fischer[♢,♣] Claudiu Musat[♡]

[♢]School of Engineering and Architecture of Fribourg, Switzerland, HES-SO, iCoSys Institute
[♠]Ecole Polytechnique Fédérale de Lausanne (EPFL) [♡]Swisscom [♣]University of Fribourg
{noemien.kocher|sciutochristian|lorenzotara7}@gmail.com
{alexandros.lazaridis|claudiu.musat}@swisscom.ch, andreas.fischer@hefr.ch

Abstract

In sequence modeling tasks the token order matters, but this information can be partially lost due to the discretization of the sequence into data points. In this paper, we study the imbalance between the way certain token pairs are included in data points and others are not. We denote this a token order imbalance (TOI) and we link the partial sequence information loss to a diminished performance of the system as a whole, both in text and speech processing tasks. We then provide a mechanism to leverage the full token order information—Alleviated TOI—by iteratively overlapping the token composition of data points. For recurrent networks, we use prime numbers for the batch size to avoid redundancies when building batches from overlapped data points. The proposed method achieved state of the art performance in both text and speech related tasks.

1 Introduction

Modeling sequences is a necessity. From time series (Connor et al., 1994; Lane and Brodley, 1999) to text (Sutskever et al., 2011) and voice (Robinson, 1994; Vinyals et al., 2012), ordered sequences account for a large part of the data we process and learn from. The data are discretized and become, in this paradigm, a list of *tokens*.

The key to processing these token sequences is to model the interactions between them. Traditionally (Rosenfeld, 2000) this has been achieved with statistical methods, like N-grams.

With the advances in computing power and the rebirth of neural networks, the dominant paradigm has become the use of recurrent neural networks (RNNs) (Mikolov et al., 2010).

The dominance of RNNs has been recently challenged with great success by self-attention based models (Vaswani et al., 2017). Instead of modeling the sequence linearly, Transformer-based models use learned correlations within the input to weight each element of the input sequence based on their relevance for the given task.

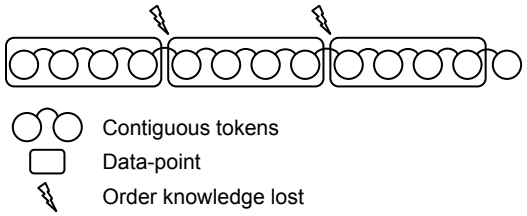


Figure 1: The common way of building data points given a dataset of contiguous tokens. Here we illustrate a dataset with a contiguous list of 13 tokens, from which we build 3 data points of 4 tokens each. This process keeps the order of the tokens inside the data points, but loses the order information from token pairs that happen to fall between adjacent data points.

Series discretization. Both RNNs and self-attention models take as input *data points*—token sequences of a maximum predefined length—and then create outputs for each of them. These tend to be much shorter in size, compared to the size of the full dataset. While for humans time seems to pass continuously, this discretization step is important for the machine understanding of the sequence.

A side effect of this step is a partial loss of the token order information. As portrayed in Figure 1, we notice that the token order information within a data point are kept. On the other hand, the knowledge about the token order at the boundaries of data points is lost. We name the situation *Token Order Imbalance (TOI)*.

As the discretization in Figure 1 is the current standard of sequence processing, we denote this as standard Token Order Imbalance (TOI). We hypothesize that this loss of information unnecessarily affects the output of the neural network models.

Alleviated Token Order Imbalance. A first contribution in this work is a mechanism to en-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 890–899
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

sure that all token sequences are taken into account, i.e. every token pair is included in a data point and does not always fall between two data point boundaries. Thus, all sequence information is available for subsequent processing. The proposed method, denoted Alleviated TOI, employs a token offset in the data point creation to create overlapped data point sequences in order to achieve this effect.

Batch Creation with Alleviated TOI. A second contribution is a strategy for batch creation when using the proposed Alleviated TOI method. We have observed an unintended data redundancy within batches introduced by the overlapped data point sequences. A strategy for avoiding this data redundancy is surprisingly simple but effective: Always use a prime number for the batch size. The intuition behind the prime batch size is that it ensures a good distribution of the batches over the entire dataset. If used naively, the Alleviated TOI policy leads to very similar data points being selected in a batch, which hinders learning. By decoupling the batch size and the token offset used in the token creation, this negative effect is effectively removed.

We then compare the Alleviated TOI with the Standard TOI and show that, on the same dataset and with the same computation allocated, the Alleviated TOI yields better results. The novel TOI reduction method is applicable to a multitude of sequence modeling tasks. We show its benefits in both text and voice processing. We employ several basic and state of the art RNNs as well as Transformers and the results are consistent—the additional information provided by the Alleviated TOI improves the final results in the studied tasks.

For text processing we focus on a well-studied task—language modeling—where capturing the sequence information is crucial. Using Alleviated TOI (P) with the Maximum Over Softmax (MoS) technique on top of a recurrent cell (Yang et al., 2017) we get the new state of the art on the Penn-Tree-Bank dataset without fine-tuning with 54.58 perplexity on the test set. We also obtain results comparable to the state of the art on speech emotion recognition on the IEMOCAP (Busso et al., 2008) dataset[1].

The paper continues with an overview of the related work in Section 2, a description of the alleviated TOI mechanism in Section 3 and a detailed description of the batch generation in Section 4. The experimental design follows in Section 5 and the results are detailed and interpreted in Section 6.

2 Related work

At the core of our work is the idea that the way that data samples are provided for training a model can affect speed or capabilities of the model. This field is broad and there are several distinct approaches to achieve it. Notable examples include curriculum learning (Bengio et al., 2009) and self-paced learning (Kumar et al., 2010), where data points for training are selected based on a metric of *easiness* or *hardness*. In Bayesian approaches (Klein et al., 2016), the goal is to create sub-samples of data points, whose traits can be extrapolated as the full dataset.

Our work thus differs from the aforementioned methods in the fact that we focus on exploiting valuable but overlooked information from sequences of tokens. We change the way data points are generated from token sequences and extend the expressivity of a model by providing an augmented, and well sorted, sequence of data points. This method has a related effect of a randomized-length backpropagation through time (BPTT) (Merity et al., 2017), which yields different data points between epochs. It also resembles classical text data-augmentation methods, such as data-augmentation using thesaurus (Zhang and LeCun, 2015).

Our method takes a step forward and proposes a systematic and deterministic approach on building data points that provides the needed variety of data points without the need of randomized-length backpropagation through time (BPTT). This has the effect of producing a text-augmentation without the need of using external resources such as a thesaurus, but only requires the dataset itself. Our method uses a concept of overlapped data points, which can be found in many areas such as data-mining (Dong and Pei, 2007), DNA sequencing (Ng, 2017), spectral analysis (Ding et al., 2000), or temporal data (Lane and Brodley, 1999). In language modeling however, this approach of overlapped data points has not yet been fully exploited. On the other hand, extracting frame-based acoustic features such as mel-fequency cepstral coefficients (MFCCs) using overlapping windows

[1]To make our results reproducible, all relevant source codes are publicly available at https://github.com/nkcr/overlap-ml

is a common technique in speech processing and more specifically in automatic speech recognition (ASR) (Chiu et al., 2018; Xiong et al., 2016; Kim and Stern, 2016). We hypothesize that extending the current overlapping technique to a higher level, that is using a sliding overlapping window over the already extracted features, will be proven beneficial. We believe this to have a positive impact on speech processing tasks such as speech emotion recognition (SER). This is because the emotional load in an spoken utterance expands over larger windows than frame-, phoneme- or syllable-based ones (Frijda, 1986).

We investigate the proposed method using a simple LSTM model and a small-size Transformer model on the IEMOCAP dataset (Busso et al., 2008), composed of five acted sessions, for a four-class emotions classification and we compare to the state of the art (Mirsamadi et al., 2017) model, a local attention based BiLSTM. Ramet et al. (2018) showed in their work a new model that is competitive to the one previously cited, following a cross-valiadation evaluation schema. For a fair comparison, in this paper we focus on a non-cross-valiation schema and thus compare our results to the work of Mirsamadi et al. (2017), where a similar schema is followed using as evaluation set the fifth session of IEMOCAP database. It is noteworthy that with a much simpler method than presented in Ramet et al. (2018), we achieve comparable results, underscoring the importance of the proposed method for this task as well.

3 Alleviated Token Order Imbalance

Let a token pair denote an ordered pair of tokens—for instance token A followed by token B, as in the sequence "$ABCDEFG...$". When splitting a token sequence into data points "$D1, D2, ..$", if the split is fixed, as in $D1$ always being equal to "ABC", $D2$ always being equal to "DEF", etc., then the information contained in the order of tokens C and D for instance is partially lost. This occurs as there is no data point that contains this token pair explicitly. We call the "CD" token pair a *split token pair* and its tokens, C and D, are denoted as *split tokens*.

In its most extreme form, split token pair order information is lost completely. In other cases, it is partially taken into account implicitly. In recurrent cells, for instance, the internal state of the cell allows for the order information of split tokens pairs to be used. This is due to the serial processing of the data points containing the split tokens.

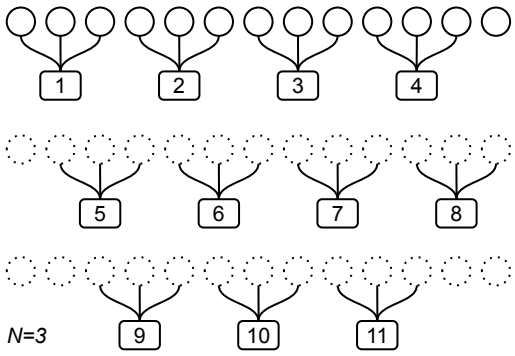


Figure 2: Illustration of an Alleviated TOI (3) made from a single contiguous list of 13 tokens. With a Standard TOI and N=3 (ie. 3 tokens per data point), a contiguous list of 13 tokens would produce 4 data points, which is illustrated by the first overlapped sequence. Here, an Alleviated TOI (3) splits the contiguous list of tokens 3 times with each time a different offset ($0, 1, 2$ respectively). This finally leads to a list of 11 data points coming from the 3 appended overlapped sequences.

As some token pairs are taken into account fully, others partially and others not at all, we denote this situation as *token order imbalance* (TOI).

In this paper, we propose to alleviate the TOI by means of overlapping sequences of data points. The aim is to avoid the loss of information between the last token of a data point and the first token of its subsequent data point. Instead of splitting the sequence of tokens only once, we repeat this process multiple times using different offsets. Each time we subdivide the sequence of tokens with a new offset, we include the links that were missing in the previous step. Finally, the overlapping sequences of data points are concatenated into a single sequence, forming the final dataset.

Figure 2 illustrates an Alleviated TOI (3), which means the sequence of data points is split three times instead of only once, producing 3 overlapped sequences that will then be concatenated.

Our Alleviated TOI (P) method is detailed in the pseudo-code below, where `olp_sequence` holds an overlapped sequence and P is the number of times we subdivide the sequence of tokens with a different offset:

```
Let N = Number of tokens per data point
    P = Number of overlapped sequences
    Step = N / P
DataPoints = empty list
FOR i = 0..P-1
```

```
  olp_sequence = create data points
                 from Dataset with
                 offset (i * Step)
  Add olp_sequence to DataPoints
RETURN DataPoints
```

When we apply an Alleviated TOI (P), this means that we are going to create P times a sequence of data points with different offsets. Therefore, the final dataset will be the concatenation of P repetitions of the original dataset, with data points shifted by a specific and increasing offset at token level for each repetition.

For example, given a sequence S_1 with $N = 70$ tokens per data point and an Alleviated TOI (P) with $P = 10$, the step size will be $\frac{N}{P} = 7$ tokens. Therefore, starting from the sequence S_1, nine additional sequences of data points will be created: S_2 starting from token 7, S_3 starting from token 14, S_4 starting from token 21 and so on until S_{10}.

When using Alleviated TOI (P), with P smaller than the data point size, within an epoch, a split token pair—that is a token pair that is split in the original data point splitting—becomes part of a data point $P - 1$ times. A token pair that is never split will be part of the data point P times.

We can thus define a *token order imbalance ratio* that describes the imbalance between the number of times we include split token pairs and the number of times we include pairs that are not split:

$$(P - 1)/P$$

We notice that the higher P, the closer the ratio becomes to 1. We hypothesize that the closer the ratio becomes to 1, the better we leverage the information in the dataset. We thus expect that for higher values of P the Alleviated TOI (P) method will outperform versions with lower values, with Alleviated TOI (1) being the Standard TOI, which is now prevalent.

We quantify the additional computational cost of Alleviated TOI (P). Since our method only results in P (shifted) repetitions of the dataset, each epoch using the augmented dataset would take $\sim P$ times longer than an epoch over the original dataset. Therefore, we ensure fair comparison by allowing baseline models to train for P times more epochs than a model using Alleviated TOI (P).

4 Batch Creation with Alleviated TOI

Series discretization may also occur at higher levels than data points, in particular when building batches for mini-batch training of neural networks. We can distinguish two types of batches, i.e. *sequential* and *distributed* batches. The former keep the data point sequences intact, thus creating split token pairs only between two consecutive batches. The latter distribute data points from different parts of the dataset to approximate the global distribution, thus creating split token pairs between all data points in batches.

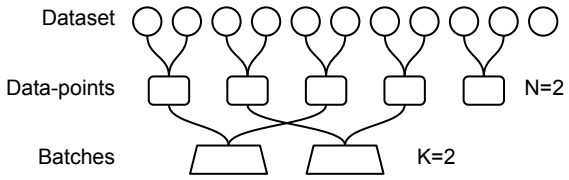


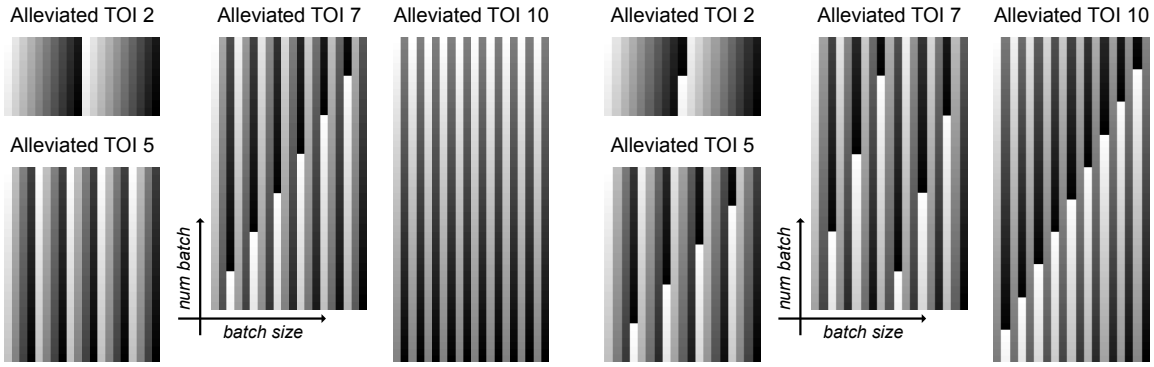
Figure 3: Three levels of data representation used to create distributed batches. The dataset is a sequence of tokens on which data points are built by splitting the sequence into subsequences of N tokens. Batches of K data points are then built by subdividing the sequence of data points into K equal parts. Here, the first part contains the first two data points, the second part the following two, and the last data point is dropped. Each batch then uses one element of each part.

In principle, our proposed method alleviates the TOI in both cases, since multiple overlapping sequences of data points are generated. However, we have observed an unintended interference with the batch creation in the case of distributed batches. In this section we explain the problem in detail and propose a simple but effective solution—choosing a prime batch size.

Figure 3 illustrates the three levels of data representation in the case of distributed batches. Data points are built from N consecutive tokens to capture the sequential information. Batches are then built from K parts of the data point sequence to capture the global distribution. An example of this approach is the batching procedure used in Zoph and Le (2016); Merity et al. (2017); Yang et al. (2017); Zołna et al. (2017) for word language modeling, where the basic token is a word.

The batching mechanism can be seen as building a 2-dimensional matrix, where each row contains a batch. Consider a sequence of M data points and a batch size of K. In order to build batches, the data points are split into K parts, represented as $\frac{M}{K} \times 1$ column vectors. They are concatenated to form a $\frac{M}{K} \times K$ matrix, such that the rows correspond to batches.

When applying the proposed Alleviated TOI (P) method (see Section 3), we augment the original



(a) Matrix of batches with batch size of 20

(b) Matrix of batches with batch size of 19

Figure 4: Illustrations of the 2D matrix of batches with different P-values of Alleviated TOI (P). On the left we used a batch size of 20 and on the right we used a prime batch size of 19. Each data point is a pixel and each row is a batch. The grayscale value models the proximity of the data points with respect to the dataset. Therefore, two pixels with similar color represents two data points that are close in the dataset. The illustrations demonstrate how different values of P affect the content of the batches, which can lack a good distribution over the dataset. Ideally, each row should contain a gradient of different grayscale values. We can observe how using a prime batch size affects the distribution of data points within the batches, where the matrices on the right offer a better distribution. This effect is especially well visible for the Alleviated TOI 10.

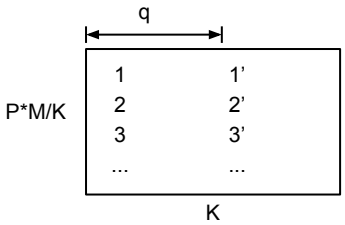


Figure 5: Data point repetition with period q for M data points, K batches, and Alleviated TOI (P). Data point 1' is the same as data point 1 with a token offset.

P	**Period** q	**Repetitions**
2	10	2
5	4	5
7	20	1
10	2	10

Table 1: Data point repetition with period q for batch size $K = 20$ and Alleviated TOI (P).

dataset to a total of $P \cdot M$ data points, adding additional data points with token offsets. Therefore, the $\frac{P \cdot M}{K} \times K$ matrix used for batch creation may contain repeated data points within the same batch as illustrated in Figure 5. A *repeated data point* differs from the previous data point only marginally due to the token offset. This redundancy can be problematic, as the batches are not well-distributed over the entire dataset anymore.

With respect to the batch matrix, a repeated data point occurs iff $\frac{P \cdot M}{K} \cdot q = n \cdot M$ with *period* $q < K$ and $q, n \in \mathbb{N}$. This is equivalent to

$$\frac{P}{K} \cdot q = n, \quad q < K, q, n \in \mathbb{N}$$

independent of the number of data points M. A repetition thus occurs iff the greatest common divisor (GCD) of P and K is larger than 1. Otherwise, for $\text{GCD}(P, K) = 1$ a data point repeats only after period $q = K$, i.e. there is no repetition within the same batch.

Table 1 lists exemplary periods for a batch size of $K = 20$ and different values of P for the Alleviated TOI (P). The worst case is $P = 10$ with 10 repetitions of the same data point within the same batch and the best case is $P = 7$, which avoids any redundancy because the GCD of P and K is 1. Figure 4 illustrates the repetition with grayscale values, where similar grayscale values indicate that two data points are close within the original data points sequence.

In general, while we aim for large values of P for reducing the TOI, a simple solution for avoiding redundancy within batches is to choose a prime number for the batch size K.

5 Experimental Setup

To validate the generalization capability of the proposed technique, we apply it on both text and

speech related tasks. We thus run the Alleviated TOI (P) with language modeling (text) and emotion recognition (speech). The text datasets used are Penn-Tree-Bank (PTB) (Marcus et al., 1993) as preprocessed in Mikolov et al. (2011), Wikitext-2 (WT2), and Wikitext-103 (WT103) (Merity et al., 2016). The speech dataset is the IEMOCAP database (Busso et al., 2008), a collection of more than 12 hours of recorded emotional speech of 10 native-English speakers, men and women. The audio data is filtered down to 5.5 hours containing only *angry*, *happy*, *neutral* and *sad* utterances.

5.1 TOI in Language Modelling

For language modeling, we use three different methods:

- A simple LSTM that does not benefit from extensive hyper-parameter optimization.
- An Average Stochastic Gradient Descent Weight-Dropped LSTM (AWD-LSTM) as described in Merity et al. (2017), with the same hyper-parameters.
- The latest State-of-the-Art model: Mixture of Softmaxes (MoS) (Yang et al., 2017).

We compare our results against the original process of building data points, i.e. Standard TOI, and use the same computation load allocated for each experiment. We use the same set of hyper-parameters as described in the base papers, except for the batch size with Alleviated TOI (P), where we use a prime batch size in order to prevent any repetitions in batches, as described in Section 4. That is, on the PTB dataset, we use a sequence length of 70 for all the models. For the Simple LSTM and AWD-LSTM, we use a batch size of 20 and a hidden size of 400. AWD-LSTM and MoS are trained on 1000 epochs, and the Simple LSTM on 100 epochs. For the MoS model, embedding size used is 280, batch size 12, and hidden size 980. All the models use SGD as the optimizer.

We set up experiments to compare 4 different token order imbalance setups: Extreme TOI, Inter-batch TOI, Standard TOI, and Alleviated TOI (P).

Extreme TOI The Extreme TOI setup builds batches using a random sequence of data points. This removes any order inside the batches (i.e. among data points within a batch), and among batches.

Inter-batch TOI In the Inter-batch TOI setup, batches are built using an ordered sequence of data points, but the sequence of batches is shuffled. This keeps the order inside batches, but removes it among batches. Looking at the 2D matrix of batches, in Figure 4, this results in shuffling the rows of the matrix.

Standard TOI In the Standard TOI setup, the process of building batches is untouched, as described in section 3. This keeps the order inside, and among batches.

Alleviated TOI (P) In the Alleviated TOI (P) setup, we apply our proposed TOI reduction by creating P overlapped data point sequences (see Sections 3 and 4). This strategy not only keeps the order inside and among batches, but it also restores the full token order information in the dataset.

5.2 TOI in Speech Emotion Recognition

For Speech Emotion Recognition (SER) we use two different models: the encoder of the Transformer (Vaswani et al., 2017) followed by convolutional layers, and the simple LSTM used in text domain case. Since the Transformer is stateless and uses self-attention instead, we are able to investigate the effect of Alleviated TOI (P) independently of LSTM cells.

As with language modeling, we set up experiments to compare the 4 different token order imbalance strategies: Extreme TOI, Inter-batch TOI, Standard TOI, and Alleviated TOI (P).

We apply the methodology used in text on the SER task, using the simple LSTM and a window size of 300 frames. In this case, a data point, instead of being a sequence of words, is a sequence of frames coming from the same utterance. Each frame is described by a 384-dimensional features vector. OpenSMILE (Eyben et al., 2013) is used for extracting the features. We opt for the IS09 features set (Schuller et al., 2009) as proposed by Ramet et al. (2018) and commonly used for SER.

Finally, to investigate the effect of the Alleviated TOI (P) strategy independently of LSTM cells, we design a final experiment in the SER task. We investigate whether or not we have improved results as we increase P, the number of overlapped data point sequences in a stateless scenario. For this reason, we use the Transformer model described above.

Experiment	PTB	WT2	WT103
Extreme TOI	63.49	73.52	36.19
Inter-batch TOI	64.20	72.61	36.39
Standard TOI	**58.94**	**65.86**	**32.94**
Alleviated TOI 2	57.97	65.14	32.98
Alleviated TOI 5	57.14	65.11	33.07
Alleviated TOI 7	57.16	64.79	32.89
Alleviated TOI 10	**56.46**	**64.73**	**32.85**

Table 2: Perplexity score (PPL) comparison of the AWD model, on the three datasets, with batch sizes $K = 20$ (PTB), $K = 80$ (WT2) and $K = 60$ (WT103), with different levels of Token Order Imbalance (TOI). With Alleviated TOI (P), we use a prime batch size of $K = 19$ (PTB), $K = 79$ (WT2) and $K = 59$ (WT103).

Model	test ppl
AWD-LSTM (Merity et al., 2017)	58.8
AWD-LSTM + Alleviated TOI	56.46
AWD-LSTM-MoS (Yang et al., 2017)	55.97
AWD-LSTM-MoS + Alleviated TOI	54.58
Simple-LSTM	75.36
Simple-LSTM + Alleviated TOI	74.44

Table 3: Comparison between state-of-the-art models (Merity et al., 2017; Yang et al., 2017) and a Simple LSTM, and the same models with Alleviated TOI. The comparison highlights how the addition of Alleviated TOI is able to improve state-of-the-art models, as well as a simple model that does not benefit from extensive hyper-parameter optimization.

6 Experimental Results

6.1 Language Modelling

Table 2 compares the 4 token order imbalance strategies using the AWD model and three text datasets. We use the test perplexity after the same equivalent number of epochs. The different Alleviated TOI (P) experiments use a different number of overlapped sequence: An Alleviated TOI (P) means building and concatenating P overlapped sequences. Our results indicate that an Alleviated TOI (P) is better than the Standard TOI, which is better than an Extreme or Inter-batch TOI. We note a tendency that higher values of P lead to better results, which is in accordance with our hypothesis that a higher TOI ratio $(P-1)/P$ improves the results.

Comparison with State of the Art and Simple LSTM. With the MoS model and an Alleviated TOI, we improve the current state of the art without fine tuning for the PTB dataset with 54.58 perplexity on the test set. Table 3 demonstrates how models can be improved by applying our Alleviated TOI method on 2 latest state-of-the-art models: AWD-LSTM (Merity et al., 2017) and AWD-LSTM-MoS (Yang et al., 2017), and the Simple LSTM model. We compare the results with the same hyper-parameters used on the original papers with the only exception of the batch size, that must be prime. To ensure fairness, we allocate the same computational resources for the base model as well the model with Alleviated TOI, i.e. we train with the equivalent number of epochs.

Experiment	K=20	K=19
Alleviated TOI 2	59.37	57.97
Alleviated TOI 5	60.50	57.14
Alleviated TOI 7	**56.70**	57.16
Alleviated TOI 10	65.88	**56.46**

Table 4: Perplexity score (PPL) comparison on the PTB dataset and the AWD model. We use two different values for the batch size K — the original one with $K = 20$, and a prime one with $K = 19$. The results directly corroborate the observation portrayed in Figure 4, where the obtained score is related to the diversity of grayscale values in each row.

Comparison without prime batch size. In Table 4 we demonstrate how using a prime batch size with Alleviated TOI (P) actually impacts the scores. We compare the scores of a prime batch size $K = 19$ with the scores of the original batch size $K = 20$ for the AWD model with Alleviated TOI (P). When using a prime batch size, we observe consistent and increasing results as P increases. This is due to the good distribution of data points in the batches regardless of the value of P, which is visible in Figure 4(b) where each row contains a high diversity of grayscale values. With the original batch size $K = 20$, we observe a strong performance for $P = 7$, but a low performance for $P = 10$. Again, this effect is related to the distribution of data points in the batches, which is visible in Figure 4(a). The matrix with $P = 7$ shows a good distribution—corresponding to the strong performance—and the matrix with $P = 10$ shows that each row contains a low diversity of data points.

Experiment	WA	UA
Extreme TOI (15k steps)	0.475	0.377
Inter-batch TOI (15k steps)	0.478	0.386
Standard TOI (15k steps)	0.486	0.404
Alleviated TOI (15k steps)	**0.553**	**0.489**
Alleviated TOI (60 epochs)	**0.591**	**0.523**

Table 5: Token order imbalance (TOI) comparison for the IEMOCAP dataset on a SER task using *angry, happy, neutral* and *sad* classes with a simple LSTM model.

Experiment	WA (60 epochs)	UA (60 epochs)
Alleviated TOI 1	0.591±0.012	0.543±0.021
Alleviated TOI 2	0.594±0.007	0.549±0.016
Alleviated TOI 3	0.605±0.018	0.563±0.024
Alleviated TOI 5	0.608±0.015	0.562±0.028
Alleviated TOI 10	**0.617±0.015**	**0.571±0.024**
Local attention	**0.635**	**0.588**

Table 6: Token order imbalance (TOI) comparison for the IEMOCAP dataset on a SER task using *angry, happy, neutral and sad* classes for 60 epochs using the Transformer model.

6.2 Speech Emotion Recognition Results

The results on the IEMOCAP database are evaluated in terms of weighted (WA) and unweighted accuracy (UA). The first metric is the accuracy on the entire evaluation dataset, while the second is the average of the accuracies of each class of the evaluation set. UA is often used when the database is unbalanced, which is true in our case, since the *happy* class has a total duration that is half of the second smallest class in speech duration.

Table 5 shows that our proposed method brings value in the speech related task as well. When choosing the Extreme TOI instead of the Standard TOI approach we observe a smaller effect than in text related task: this is due to the different nature of the text datasets (large "continuous" corpuses) and the IEMOCAP one (composed of shorter utterances). The fact that we can still observe improvements on a dataset with short utterances is a proof of the robustness of the method.

A greater effect is obtained when we increase the size of the dataset with the proposed Alleviated TOI (P) approach: Due to the increasing offset at each overlapped sequence, the data fed into the model contains utterances where the emotions are expressed in slightly different ways. For this reason, the performance notably increases.

Table 6 reports the result of a final experiment that aims to investigate the effect of Alleviated TOI (P) independently of LSTM cells. For each Alleviated TOI (P) setup and Standard TOI described in Table 6, we repeat the training and evaluation for each of the following window sizes: 100, 200, 300, 400 and 500 frames. The previously described Transformer model is used in these experiments. The results reported in Table 6 are the mean $\pm$ the standard deviation computed for different P-values of Alleviated TOI (P).

The last line of Table 6 refers to Mirsamadi et al. (2017) results. We want to highlight the fact that the goal of these experiments is to show the direct contribution of the Alleviated TOI technique for a different model. For this reason we use a smaller version of the Transformer in order to reduce the computational cost. We believe that with a more expressive model and more repetitions, the proposed method may further improve the results.

The results from Table 6 demonstrate that, as we increase the value of P, more significant improvements are achieved. This is in accordance with our hypothesis that a higher TOI ratio $(P-1)/P$ improves the results.

7 Conclusions

In this work, the importance of overlapping and token order in sequence modelling tasks were investigated. Series discretization is an essential step in machine learning processes which nonetheless can be responsible for the loss of the continuation of the tokens, through the token order imbalance (TOI) phenomenon. The proposed method, Alleviated TOI, has managed to overcome this drawback and ensures that all token sequences are taken into account. The proposed method was validated in sequence modelling tasks both in the text and speech domain outperforming the state of the art techniques.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower Provost, Samuel Kim, Jeannette N. Chang, Sungbok Lee, and Shrikanth

Narayanan. 2008. Iemocap: interactive emotional dyadic motion capture database. *Language Resources and Evaluation*, 42:335–359.

Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al. 2018. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4774–4778. IEEE.

Jerome T Connor, R Douglas Martin, and Les E Atlas. 1994. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254.

Mingzhou Ding, Steven L Bressler, Weiming Yang, and Hualou Liang. 2000. Short-window spectral analysis of cortical event-related potentials by adaptive multivariate autoregressive modeling: data preprocessing, model validation, and variability assessment. *Biological cybernetics*, 83(1):35–45.

Guozhu Dong and Jian Pei. 2007. *Sequence data mining*, volume 33. Springer Science & Business Media.

Florian Eyben, Felix Weninger, Florian Gross, and Björn Schuller. 2013. Recent developments in opensmile, the munich open-source multimedia feature extractor. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 835–838, New York, NY, USA. ACM.

Nico H. Frijda. 1986. *The Emotions*. Cambridge University Press.

Chanwoo Kim and Richard M Stern. 2016. Power-normalized cepstral coefficients (pncc) for robust speech recognition. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(7):1315–1329.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. 2016. Fast bayesian optimization of machine learning hyperparameters on large datasets. *CoRR*, abs/1605.07079.

M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.

Terran Lane and Carla E Brodley. 1999. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security (TISSEC)*, 2(3):295–331.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *CoRR*, abs/1708.02182.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukáš Burget, and Jan Černockỳ. 2011. Empirical evaluation and combination of advanced language modeling techniques. In *Twelfth Annual Conference of the International Speech Communication Association*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Seyedmahdad Mirsamadi, Emad Barsoum, and Cha Zhang. 2017. Automatic speech emotion recognition using recurrent neural networks with local attention. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2227–2231.

Patrick Ng. 2017. dna2vec: Consistent vector representations of variable-length k-mers. *arXiv preprint arXiv:1701.06279*.

Gaetan Ramet, Philip N. Garner, Michael Baeriswyl, and Alexandros Lazaridis. 2018. Context-aware attention mechanism for speech emotion recognition. *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 126–131.

Anthony J. Robinson. 1994. An application of recurrent nets to phone probability estimation. *Trans. Neur. Netw.*, 5(2):298–305.

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278.

Björn Schuller, Stefan Steidl, and Anton Batliner. 2009. The interspeech 2009 emotion challenge. In *Tenth Annual Conference of the International Speech Communication Association*.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Oriol Vinyals, Suman V. Ravuri, and Daniel Povey. 2012. Revisiting recurrent neural networks for robust asr. In *2012 IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP), pages 4085–4088.

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. 2016. Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W. Cohen. 2017. Breaking the softmax bottleneck: A high-rank RNN language model. *CoRR*, abs/1711.03953.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Konrad Zołna, Devansh Arpit, Dendi Suhubdy, and Yoshua Bengio. 2017. Fraternal dropout. *stat*, 1050:31.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

Global Autoregressive Models for Data-Efficient Sequence Learning

Tetiana Parshakova
Stanford University*
tetianap@stanford.edu

Jean-Marc Andreoli
Naver Labs Europe

Marc Dymetman
Naver Labs Europe

{jean-marc.andreoli,marc.dymetman}@naverlabs.com

Abstract

Standard autoregressive seq2seq models are easily trained by max-likelihood, but tend to show poor results under small-data conditions. We introduce a class of seq2seq models, GAMs (Global Autoregressive Models), which combine an autoregressive component with a log-linear component, allowing the use of global *a priori* features to compensate for lack of data. We train these models in two steps. In the first step, we obtain an *unnormalized* GAM that maximizes the likelihood of the data, but is improper for fast inference or evaluation. In the second step, we use this GAM to train (by distillation) a second autoregressive model that approximates the *normalized* distribution associated with the GAM, and can be used for fast inference and evaluation. Our experiments focus on language modelling under synthetic conditions and show a strong perplexity reduction of using the second autoregressive model over the standard one.

1 Introduction

Neural sequential text generation models have become the standard in NLP applications such as language modelling, NLG, machine translation. When enough data is available, these models can be trained end-to-end with impressive results. Generally, inference and training proceed in an auto-regressive manner, namely, the next decoded symbol is predicted by a *locally normalized* conditional distribution (the "softmax"). This has several advantages: (i) the probability of the sequence is already normalized, by the chain-rule over local decisions, (ii) max-likelihood (ML) training is easy, because the log-likelihood of the full sequence is simply the sum of local CE (cross-entropy) losses, (iii) exact sampling of full sequences from the model distribution is directly obtained through a sequence of local sampling decisions.

However, these autoregressive models (AMs) tend to suffer from a form of myopia. They have difficulty accounting for global properties of the predicted sequences, from overlooking certain aspects of the semantic input in NLG to duplicating linguistic material or producing "hallucinations" in MT, and generally through being unable to account for long-distance consistency requirements that would be obvious for a human reader.[1]

The main contributions of this paper are as follows.

First, we propose a hybrid seq2seq formalization, the *Global Autoregressive Model* (GAM), that combines a local autoregressive component with a global log-linear component, allowing the use of *a priori* features to compensate for the lack of training data. GAMs are related both to the class of Energy-Based Models (EBM) and to that of Exponential Families (EF), and inherit some important properties from those: an intimate relationship between training and sampling (EBM); the identity of empirical and model expectations at maximum-likelihood; convexity of log-likelihood (EF).

Second, we propose a training procedure in two steps. In the first step, we train through max-likelihood a GAM, which however is *unnormalized* and improper for fast inference or evaluation. In the second step, we use this GAM to train (by distillation) a second autoregressive model that approximates the *normalized* distribution associated with the GAM, and can be used for fast inference

* Work conducted during an internship at NAVER Labs Europe.

[1] To borrow terminology from Reinforcement Learning (RL) (Sutton and Barto, 2018), such NLP models work by "imitation learning", without any representation of "objectives" to be realized. While this defect can be mitigated in the presence of large training sets, it can become serious when this condition is not met.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 900–909
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

and evaluation.

Third, we demonstrate the ability of GAMs to be data-efficient, namely, to exploit the original data better than a standard autoregressive model. In order to clarify the core techniques and issues, we design a simple class of synthetic data, consisting of random binary strings containing "motifs" (specific substrings) that we can manipulate in different ways. We show that, in limited data conditions, GAMs are able to exploit the features to obtain final autoregressive models that perform better than the original ones.

The remainder of the paper is structured as follows. In Section 2, we provide some background about autoregressive models, energy-based models, and log-linear models. In Section 3, we introduce GAMs. In section 4, we describe our focus on synthetic data. In Section 5, we explain our training procedure. In Section 6, we comment on related work. In Section 7, we describe our experiments. In Section 8, we provide an analysis of our results. We conclude with a discussion in Section 9. *Note that some additional explanations and experiments are provided in the Supplementary Material, indicated by [SM].*

2 Background

2.1 Autoregressive models (AM)

These are currently the standard for neural seq2seq processing, with such representatives as RNN/LSTMs (Hochreiter and Schmidhuber, 1997; Sutskever et al., 2014), ConvS2S (Gehring et al., 2017), Transformer (Vaswani et al., 2017)). Formally, they are defined though a distribution $r_\eta(x|C)$, where C is an input (aka Context, e.g. a source sentence in Machine Translation (MT)), and x is a target sequence (e.g. a target sentence in MT). We have:

$$r_\eta(x|C) \doteq \prod_i s_\eta(x_i|x_1, \ldots, x_{i-1}, C),$$

where each $s_\eta(x_i|x_1, \ldots, x_{i-1}, C)$ is a normalized conditional probability over the next symbol of the sequence, computed by a neural network (NN) with parameters η. The local normalization of the incremental probabilities implies the overall normalization of the distribution $r_\eta(x|C)$, and consequently, the possibility of directly sampling from it and evaluating the likelihood of training sequences.

2.2 Energy-Based Models (EBM)

EBMs are a generic class of models, characterized by an energy function $U_\eta(x|C)$ computed by a NN parametrized by η (LeCun et al., 2006). Equivalently, they can be seen as directly defining a potential (an unnormalized probability distribution) $P_\eta(x|C) = e^{-U_\eta(x|C)}$, and indirectly the normalized distribution $p_\eta(x|C) = 1/Z_\eta(C)\ P_\eta(x|C)$, with $Z_\eta(C) = \sum_x P_\eta(x|C)$. A fundamental property of these models is that, for max-likelihood training, the SGD updates can be computed through the formula:[2]

$$\nabla_\eta \log p_\eta(x|C) = \nabla_\eta \log P_\eta(x|C) - E_{x\sim p_\eta(\cdot|C)} \nabla_\eta \log P_\eta(x|C), \quad (1)$$

which, in principle, reduces the problem of *training* with unnormalized potentials to the problem of *sampling* from them.

2.3 Log-Linear Models / Exponential Families

Log-Linear models (Jebara, 2013) are the conditional version of Exponential Families (Jordan, 2010). The general form of a log-linear model (for the discrete case) is as follows:

$$p_\lambda(x|C) = 1/Z_\lambda(C)\ \mu(x;C)\ e^{\langle \lambda(C),\ \phi(x;C)\rangle},$$

with $Z_\lambda(C) = \sum_x \mu(x;C)\ e^{\langle \lambda(C),\ \phi(x;C)\rangle}$. Here $\phi(x;C)$ is a vector of predefined real features of the pair (x, C), which is combined by scalar product with a real vector of weights $\lambda(C)$ of the same dimension; $\mu(x;C)$ is an arbitrary "base measure", which is fixed. These models, which allow to introduce prior knowledge through features and have nice formal properties (see below), were mainstream in NLP before the revival of neural approaches.

3 Proposal: GAMs

We now define *Global Autoregressive Models* (GAMs). These are hybrid seq2seq models that exploit both local autoregressive properties as well as global properties of the full target sequence. A GAM is an unnormalized distribution $P_\eta(x|C)$ over sequences x, parametrized by a vector $\eta = \eta_1 \oplus \eta_2$:

$$P_\eta(x|C) = r_{\eta_1}(x|C) \cdot e^{\langle \lambda_{\eta_2}(C),\ \phi(x;C)\rangle}. \quad (2)$$

[2] See (LeCun et al., 2006, p. 15), and [SM] for a derivation.

Here $r_{\eta_1}(x|C)$ is an autoregressive seq2seq model for generating x from input C, parametrized by η_2; $\phi(x;C)$ is a vector of predefined real features of the pair (x, C), which is combined by a scalar product with a real vector $\lambda_{\eta_2}(C)$ of the same dimension, computed over the input C by a network parametrized by η_2. The normalized distribution associated with the GAM is $p_\eta(x|C) = \frac{P_\eta(x|C)}{Z_\eta(C)}$, where $Z_\eta(C) = \sum_x P_\eta(x|C)$.

GAMs appear promising for the following reasons:

- Features $\phi(x;C)$ provide a simple way to draw attention of the model to potentially useful aspects that may be difficult for the AM component to discover on its own from limited data.

- GAMs are an instance of EBMs, where the potential $P_\eta(x|C)$ is the product of the an AM potential $r_{\eta_1}(x|C)$ with a "log-linear" potential $e^{\langle \lambda_{\eta_2}(C), \phi(x;C) \rangle}$. Here the gradient relative to the log-linear part takes the especially simple form:

$$\nabla_{\eta_2} \log p_\eta(x|C) = \phi(x;C) - E_{x \sim p_\eta(\cdot|C)}\, \phi(x;C). \quad (3)$$

- Log-linear models, on their own, while great at expressing prior knowledge, are not as good as AM models at discovering unforeseen regularities in the data. Also, they are typically problematic to train from a log-likelihood perspective, because sampling from them is often unfeasible. GAMs address the first issue through the r component, and alleviate the second issue by permitting the use of r as a powerful "proposal" (aka "surrogate") distribution in importance sampling and related approaches, as we will see.

4 Experimental focus

While the motivation for GAMs ultimately lies in practical NLP applications such as those evoked earlier, in this paper we aim to understand some of their capabilities and training techniques in simple and controllable conditions. We focus on the unconditional (i.e. language modelling) case, and on synthetic data. Our setup is as follows:

- We consider an underlying process p_{true} that generates *binary sequences* according to a well-defined and flexible process. In this paper we use PFSAs (Probabilistic Finite State Automata) to impose the presence or absence of sub-strings ("motifs") anywhere in the generated data, exploiting the intersection properties of automata.

- Due to the dynamic programming properties of PFSAs, it is possible to compute the true entropy $H(p_{true}) = -\sum_x p_{true}(x) \log p_{true}(x)$ of the process (see [SM]), as well as other quantities (Partition Functions, Mean sequence length); it is also possible to generate training (D), validation (V), and test data (T) in arbitrary quantities.

- We employ an unconditional GAM of the simple form:

$$p_\lambda(x) \doteq \frac{P_\lambda(x)}{Z_\lambda}, \text{with } Z_\lambda \doteq \sum_x P_\lambda(x) \text{ and } P_\lambda(x) \doteq r(x) \cdot e^{\langle \lambda,\, \phi(x) \rangle}, \quad (4)$$

where r is trained on D and then kept fixed, and where λ is then trained on top of r, also on D.

It should be noted that with r fixed in this way, this formulation exactly corresponds to the definition of an exponential family (Jordan, 2010), with r as *base measure*. In such models, we have two important properties: (i) the log-likelihood of the data is *convex* relative to the parameters λ, and thus a local maximum is also global; (ii) the max-likelihood value λ^* has the property that the model expectation $E_{x \sim p_{\lambda^*}(\cdot)}\, \phi(x)$ is equal to the empirical expectation $|D|^{-1} \sum_{x \in D} \phi(x)$ ("Moment Matching" property of exponential families).

- We are specially interested in the relative *data-efficiency* of the GAM compared to the AM r: namely the ability of the GAM to recover a lower perplexity approximation of p_{true} than r, especially in small training-set conditions.

5 Training procedure

5.1 Two-stage training

We consider a two-stage training procedure (see Fig. 1).

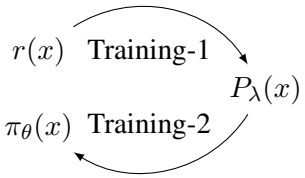


Figure 1: Two-stage training. At the end of the process, we compare the perplexities of r and π_θ on test data: $CE(T, r)$ vs. $CE(T, \pi_\theta)$.

Training-1 This consists in training the model P_λ on D. This is done by first training r on D in the standard way (by cross-entropy) and then by training λ by SGD with the formula (adapted from (3)):

$$\nabla_\lambda \log p_\lambda(x) = \phi(x) - E_{x \sim p_\lambda(\cdot)}\, \phi(x). \quad (5)$$

The main difficulty then consists in computing an estimate of the model moments $E_{x \sim p_\lambda(\cdot)}\, \phi(x)$. In our experiments, we compare two Monte-Carlo approaches (Robert and Casella, 2005) for addressing this problem: (i) *Rejection Sampling* (rs), using r as the proposal distribution and (ii) *Self-Normalized Importance Sampling* (snis) (Owen, 2017; Y. Bengio and J. S. Senecal, 2008), also using r as the proposal.

Rejection sampling is performed as follows. We use $r(x)$ as the proposal, and $P_\lambda(x) = r(x)\ e^{\lambda \cdot \phi(x)}$ as the unnormalized target distribution; for any specific λ, because our features are bounded between 0 and 1, we can easily upper-bound the ratio $\frac{P_\lambda(x)}{r(x)} = e^{\lambda \cdot \phi(x)}$ by a number β; we then sample x from r, compute the ratio $\rho(x) = \frac{P_\lambda(x)}{\beta\ r(x)} \leq 1$, and accept x with probability $\rho(x)$. The accepted samples are unbiased samples from $p_\lambda(x)$ and can be used to estimate model moments.

Snis also uses the proposal distribution r, but does not require an upper-bound, and is directly oriented towards the computation of expectations. In this case, we sample a number of points $x_1, \ldots, x_N$ from r, compute "importance ratios" $w(x_i) = \frac{P_\lambda(x_i)}{r(x_i)}$, and estimate $E_{x \sim p_\lambda(\cdot)}\, \phi(x)$ through $\hat{E} = \frac{\sum_i w(x_i)\phi(x_i)}{\sum_i w(x_i)}$. The estimate is biased for a given N, but consistent (that is, it converges to the true E for $N \to \infty$).

Training-2 While Training-1 results in a well-defined model $P_\lambda(x)$, which may fit the data closely in principle, we should not conclude that $P_\lambda(x)$ is convenient to use for inference — namely, in language modeling, efficiently sampling from its normalized version $p_\lambda(x)$; as seriously, because of the partition factor Z_λ, it is also not obvious to *evaluate* the perplexity of $P_\lambda(x)$ on test data. In order to do both, one approach consists in using a *distillation* technique (Hinton et al., 2015), where, during training, one expends generous time towards producing a set of samples from P_λ, for instance by Monte-Carlo (e.g. Rejection Sampling) techniques, and where this set (which may be arbitrarily larger than the original D) is in turn used to train a new autoregressive model $\pi_\theta(x)$, which can then be used directly for sampling or for computing data likelihood. This is the approach that we use in our current experiments, again using the original $r(x)$ as a proposal distribution.

5.2 Cyclical training

In the case of small $|D|$, the proposal distribution r is weak and as a result the distillation process, based on rejection sampling, can be slow. To address this issue, we also consider a cyclical training regime that updates the proposal distribution after distilling each batch of samples, with the intention of reducing the rejection rate. Once the process of distillation is finished, we use the aggregated samples to train the final π_θ. The two-stage training procedure is a variant of the cyclical one, with a fixed proposal (see Algorithm 1 for more details).

6 Related Work

(Hoang et al., 2018), working in a NMT context, have a similar motivation to ours. They first train an autoregressive seq2seq model (Transformer in their case) on bilingual data, then attempt to control global properties of the generated sequences through the introduction of a priori features. They interpolate the training of the autoregressive model with training of a Moment Matching component which tries to equate the features expectations of the model with those of the data. Contrarily to our approach, they do not directly try to maximize likelihood in an integrated model.

(Andor et al., 2016) consider transition-based neural networks, and contrast local to global normalization of decision sequences, showing how the global approach avoids the *label bias* problem in such tasks as tagging or parsing. They

Algorithm 1 Training

function TRAIN(D, V, T, ft, DsSize, tReg, mode)
  $r \leftarrow$ TRAINRNN(D, V, optAdam) ▷ initialize and then train RNN
  $P_\lambda \leftarrow$ TRAINGAM(r, D, V, tReg, ft) ▷ train λ for a given proposal r
  if mode = 'two_stage' **then** ▷ Training-2: distill in one step
    $\widetilde{D}$, $\widetilde{V}$, _ $\leftarrow$ DISTILLBATCH(P_λ, DsSize)
  else if mode = 'cyclic' **then** ▷ Cyclic-training: distill in several steps
    $\widetilde{D} \leftarrow \{\}$; $\widetilde{V} \leftarrow \{\}$; flag$_\lambda \leftarrow$ False
    while $|\widetilde{D}| <$ DsSize **do** ▷ proceed to the distillation process
      $\widetilde{D}_B$, $\widetilde{V}_B$, accptRate $\leftarrow$ DISTILLBATCH(P_λ, bSize) ▷ accptRate - acceptance rate of rs during distillation
      $\widetilde{D}$.insert($\widetilde{D}_B$); $\widetilde{V}$.insert($\widetilde{V}_B$)
      if not flag$_\lambda$ **then**
        $r \leftarrow$ SINGLEUPDATERNN(r, $\widetilde{D}_B$, optAdam) ▷ improve proposal r
        $P_\lambda \leftarrow$ TRAINGAM(r, D, V, tReg, ft) ▷ train λ for a given proposal r
        flag$_\lambda \leftarrow$ EARLYSTOPPING$_d$(acceptRate) ▷ check if acceptance rate has stopped improving
    $\widetilde{D}$.insert(D); $\widetilde{V}$.insert(V) ▷ add true data to the distilled one
  $\pi_\theta \leftarrow$ TRAINRNN($\widetilde{D}$, $\widetilde{V}$, optAdam)
  return π_θ
function TRAINGAM(P_λ, D, V, tReg, ft) ▷ Training-1
  $\alpha_0 \leftarrow 10$ ▷ initial learning rate
  target_mom $\leftarrow$GETMOMENTS(D, V, ft) ▷ empirical moments of the given dataset
  while not EARLYSTOPPING(ℓ_1_mom) **do** ▷ check if ℓ_1_mom has stopped improving
    model_mom $\leftarrow [0] \times |ft|$ ▷ accumulate the model's moments
    $\alpha_t \leftarrow \frac{\alpha_0}{1+\#\text{epoch}}$
    for $b \in$ range(#updatesPerEpoch) **do**
      mean_mom $\leftarrow$ GETMOMENTSGAM(P_λ, D, V, tReg, ft) ▷ use rs or $snis$ to estimate $E_{x\sim p_\lambda(\cdot)}\,\phi(x)$
      model_mom $\leftarrow$ (model_mom + mean_mom$/(b-1)) \cdot \frac{b-1}{b}$ ▷ moving average
      $\nabla_\lambda \leftarrow$ target_mom $-$ mean_mom ▷ use Eq. 5 to compute gradients
      $\lambda \leftarrow \lambda + \alpha_t \cdot \nabla_\lambda$
    ℓ_1_mom $\leftarrow \|$target_mom $-$ model_mom$\|_1$
  return P_λ

focus on inference as maximization, e.g. finding the best sequence of tags for a sequence of words, and consistent with that objective, their training procedure exploits a beam-search approximation. By contrast, our focus is on inference as sampling in a language modelling perspective, on the complementarity between auto-regressive models and log-linear models, and on the relations between training and sampling in energy-based models.

7 Experiments

We conduct a series of experiments on synthetic data to illustrate our approach.

7.1 Synthetic data

To assess the impact of GAMs, we focus on distributions $p_{true}(x)$ that are likely to be well approximated by the AM $r(x)$ in the presence of large data. The first class of distributions is obtained through a PFSA that filters binary strings of fixed length $n = 30$, 0's and 1's being equally probable (white-noise strings), through the condition that they contain a specific substring ("motif") anywhere; here the relative frequency of sequences containing the motif among all sequences varies from ~ 0.01 (shorter motifs $|m| = 10$) to ~ 0.001 (longer motifs $|m| = 14$).

We also consider mixtures of two PFSAs (motif/anti-motif): the first (with mixture prob. 0.9) produces white-noise strings containing the motif and the second (with mixture prob. 0.1) strings excluding the motif.

From these processes we produce a training set D, of size $|D|$ varying between $5 \cdot 10^2$ and $2 \cdot 10^4$, a validation set V of size $0.25 \cdot |D|$ (but never smaller than $5 \cdot 10^2$ or bigger than $2 \cdot 10^3$) and a test set T of fixed size $5 \cdot 10^3$.

7.2 Features

In a real world scenario, prior knowledge about the true process will involve, along with predictive features, a number of noisy and useless features. By training the λ parameters to match the empirical moments, the GAM will learn to distinguish between these types. In order to simulate this situation we consider feature vectors over our artificial data that involve both types.

With x the full string and m the fixed motif used in constructing the training data, we consider variations among the 7 binary features in the set F:

$$F = \{m,\ m_{+0},\ m_{/2},\ d_0,\ d_1,\ d_2,\ d_3\},$$

where $m = 0$ iff the motif m appears in x, $m_{+0} = 0$ iff the motif followed by a zero ("super-motif") appears in x, $m_{/2} = 0$ iff an initial section of the motif ("sub-motif", roughly half the size of m) appears in x. These three features are chosen because they have some correlation with

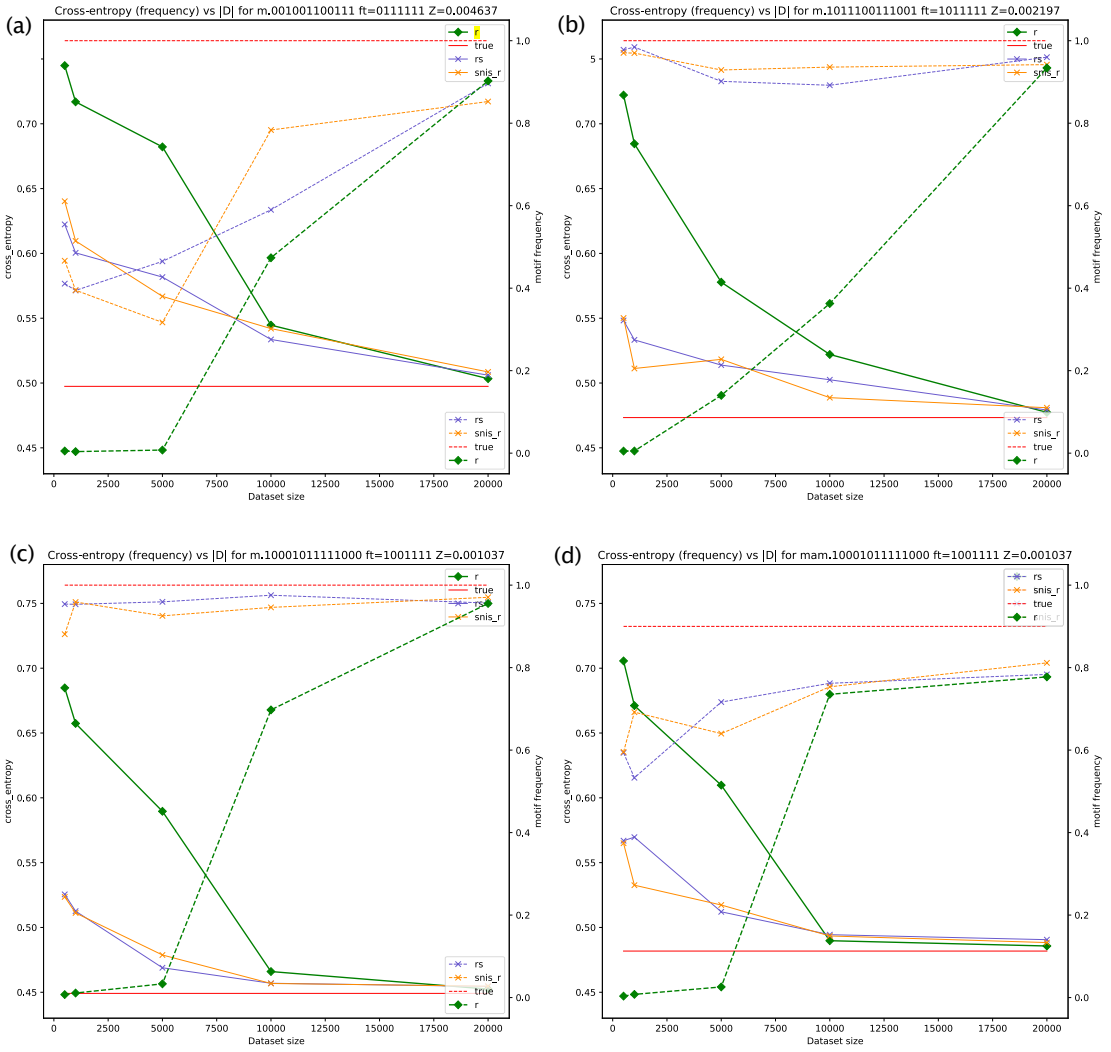


Figure 2: Cross-entropy in nats per character and frequency of sampling motif, depending on $|D|$. Two-stage Training. Features d_0, d_1, d_2, d_3 are on for all panels ($ft_{[4:7]} = \{1111\}$). Panel (a): pure D, features m_{+0} (super-motif) and $m_{/2}$ (sub-motif) on; (b): pure D, m (motif) and $m_{/2}$ (sub-motif) on; (c) pure D, m on; (d) mixture D, m on. The plain lines represent cross-entropy, the dashed lines motif frequency.

the process for generating the training data. By contrast, the four remaining features are "distractors": $d_0 = 0$ iff x begins with a 0, $d_1 = 0$ (resp. $d_2 = 0$, $d_3 = 0$) iff a certain random, but fixed, string of similar length to m (resp. of larger length, of smaller length) appears in x. We test different configurations of these features for training λ, and document the use/non-use of features with a bit-vector ft of length $|F|$, for instance $ft = 0111111$ means that all features are exploited, apart from m.[3]

[3]In the experiments reported here, one of the provided features, m, is a detector of the motif actually present in the data generating process, an extreme form of prior knowledge used to illustrate the technique. In general, milder forms of useful prior features can be provided. A simple formal example is to consider one real-valued (non binary) feature for the length, and one for the square of the length, an experiment that we did recently but do not report here; by matching the data expectations of these two additional features, the model is able to represent the mean and variance of length in the data. Here the prior knowledge provided to the model just tells it to be attentive to the distribution of length, a much weaker form of prior knowledge than telling it to be attentive to a specific motif.

7.3 Implementation aspects

7.3.1 Autoregressive models

The AMs are implemented in PyTorch[4] (Paszke et al., 2017) using a 2-layered LSTM (Hochreiter and Schmidhuber, 1997) with hidden-state size 200. The input is presented through one-hot encodings over the vocabulary $V = \{0, 1, \langle \text{EOS} \rangle\}$. These LSTMs are optimized with Adam (Kingma and Ba, 2014), with learning rate $\alpha = 0.001$, and

[4]https://github.com/parshakova/GAMS-for-Data-Efficient-Learning

with early stopping (patience $= 20$) over a validation set.

7.3.2 Training: Two-Stage and Cyclical

The implementation is described in (Algorithm 1). Here we provide some additional details.

Training-1 For training $P_\lambda(x)$ we test two regimes in Eq. 5, namely rs and $snis$; in both cases, we first train $r(x)$ on the whatever D is available, and use it as the proposal distribution. During rs, we compute the model's expectation over 10 accepted samples, update the λ's according to (5), and iterate. During $snis$, we keep a buffer of the last $5 \cdot 10^4$ samples from $r(x)$ to compute the weighted average of the feature moments. For the training of λ's, we use a basic SGD optimization with learning rate $\alpha(\text{\#epoch}) = \frac{\alpha_0}{1+\text{\#epoch}}, \alpha_0 = 10$. To assess the quality of $P_\lambda(x)$ for early stopping during training, we use the distance between the empirical and model moments:

$$\ell_1\text{_mom} = \left\| \frac{1}{|D|} \sum_{d \in D} \phi(d) - E_{x \sim p_\lambda(\cdot)}\, \phi(x) \right\|_1 . \quad (6)$$

Training-2 and Cyclical Training When distilling from P_λ in Training-2, we use a single proposal r, and systematically produce a distilled dataset of size DsSize $= 2 \cdot 10^4$, which corresponds to the highest value of $|D|$ among those considered for training r. In Cyclical Training, the distillation process is performed in several stages, with an evolving r for improving the rejection rate.

8 Results

8.1 Cross-entropy comparison

We conduct experiments to compare the cross-entropy (measured in nats) between the initial AM $r(x)$ relative to the test set T and the final AM $\pi_\theta(x)$ also relative to T; we vary the size of $|D| \in \{0.5, 1, 5, 10, 20\} \cdot 10^3$, the regimes (tReg) for Training-1 (rs or $snis$), the features employed, the rarity of the motifs. Figure 2 depicts the resulting curves at the end of the two-stage training (plain lines).

Here we show only a few experiments (a more extensive set is provided in the [SM]).

We observe that, for a small dataset size $|D|$, there is a big gap between the CE of $r(x)$ and the CE of $\pi_\theta(x)$. As $|D|$ increases, these cross-entropies become closer to one another, but a large gap persists for $|D| = 5000$.

We note that the presence of the "fully-predictive" feature m results in a $\pi_\theta(x)$ that has CE very close to the theoretical entropy, even in low $|D|$ regimes, where r on its own is very weak.[5] Thus, not only is the distilled AM much better than the initial AM, but this is an indication that P_λ itself (for which the cross-entropy is more difficult to compute exactly) is a good approximation of the true process.

By contrast, if the m feature is absent, then, while π_θ is still better than r in low $|D|$ regimes, it cannot reach the theoretical entropy in such regimes, because features such as m_{0+} and $m_{/2}$ can only partially model the data. With large $|D|$, on the other hand, r on itself does a good job at predicting the data, and P_λ adds little on top of its r component.

Finally, we note that the two regimes for training $P_\lambda(x)$, rs and $snis$, result in π_θ's with similar accuracies.

We also observe that with a good performance of $\pi_\theta(x)$, the moments of motif feature on the distilled dataset are close to the true ones (see [SM] Figure 4, 5, 7).

These trends are consistent across the experiments with different motifs, as can be checked in Table 3 and with the additional plots in the [SM].

8.2 Motif frequencies

In order to assess the *predictive* properties of obtained AMs, we also compare the frequency of motifs in strings sampled from r and from π_θ ($2 \cdot 10^3$ samples in total). From Figure 2 we see that when vary $|D|$, the frequency of motifs (dashed lines) is aligned with the CE performance. Namely, π_θ produces a higher fraction of strings with motif than r when $|D|$ is small ($|D| \in \{0.5, 1, 5\} \cdot 10^3$).

Detailed illustration To provide more intuition, we provide an illustration from one experiment in Table 1.

8.3 Mixture D_{mam} vs pure D_m

In our experiments, the strings in D_{mam} (motif-anti-motif) contain a motif with $p = 0.9$. However, if not all of the samples in D_{mam} contain the

[5] The CE of a model relative to the true underlying process (approximated by the test set T) can never be below the entropy of this process, due to the KL-divergence being non-negative.

1	$true$	101**10001011111000**1000001001001
2	r	011111000010111110001110001011
3	π_θ	111010**10001011111000**0111111100
4	ft	$[m, _, _, d_0, d_1, d_2, d_3]$
5	λ's	$[-10.1, _, _, -0.15, -0.06, 0.0, -0.14]$
6	mom $true$	$[0.0, _, _, 0.47, 0.99, 1.0, 0.91]$
7	mom r	$[0.95, _, _, 0.53, 0.99, 1.0, 0.91]$
8	mom π_θ	$[0.0006, _, _, 0.43, 0.99, 0.99, 0.91]$
9	CEs	$true$: 0.45, r: 0.56, π_θ: 0.47
10	motif freqs	$true$: 1.0, r: 0.045, π_θ: 0.959

Table 1: Illustration. Setting is from Fig. 2, panel (c): n =30, motif = 10001011111000 (always present in D), ft = 1001111, $|D|$ = 5000, rs used for Training-1. Lines 1,2,3 show one example from $true, r, \pi_\theta$ respectively; with training set of size 5000, r is only able to generate the motif a fraction of the time (0.045, see line 10), but is better able to generate some submotifs (underlined); π_θ generates the motif frequently (0.959), as illustrated on line 3. With the features from ft (line 4), Training-1 produces a P_λ with first feature λ_m strongly negative (line 5), meaning that P_λ strongly penalizes the absence of the motif; the "distractor" features d_0, d_1, d_2, d_3 get a weight close to 0, meaning that they have little predictive power in combination with feature m. It is visible from lines 6,7,8 that π_θ is much better able to approximate the true feature expectations than r [features expectations (aka moments) under r (resp. π_θ) : $E_{x\sim r(\cdot)}\ \phi(x)$ (resp. $E_{x\sim \pi_\theta(\cdot)}\ \phi(x)$)] Finally (line 9), the CE of π_θ relative to the test set is close to the true entropy of the process, while that of r is much further away.

$\lvert D\rvert$	m; $\frac{\text{mtf_frq}_{rs}}{\text{mtf_frq}_{snis}}$	m; $\frac{\text{CE}(rs)}{\text{CE}(snis)}$	m; $\frac{\text{time}(rs)}{\text{time}(snis)}$	mam; $\frac{\text{mtf_frq}_{rs}}{\text{mtf_frq}_{snis}}$	mam; $\frac{\text{CE}(rs)}{\text{CE}(snis)}$	mam; $\frac{\text{time}(rs)}{\text{time}(snis)}$
500	0.998	0.967	2.92	0.997	1.003	4.7
1000	1.009	0.973	2.038	0.77	1.07	3.638
5000	0.995	0.967	0.756	1.12	0.99	1.365
10000	1.134	0.956	1.514	1.011	1.002	1.005
20000	1.497	0.961	0.938	0.965	1.005	0.975

Table 2: Comparison of the time for Training-1 in rs and $snis$; for motif 10001011111000; $ft = 1011111$; $H(p_{true}) = 0.449$ with pure D (m) and $ft = 1001111$; $H(p_{true}) = 0.482$ with mixture of motif-anti-motif D (mam).

tReg	$\lvert D\rvert$	m: $\frac{\text{CE}(T,r)}{\text{CE}(T,\pi_\theta)}$	m: $\frac{\text{CE}(T,\pi_\theta)}{H(p_{true})}$	m: $\frac{\text{mtf_frq}(\pi_\theta)}{\text{mtf_frq}(r)}$	mam: $\frac{\text{CE}(T,r)}{\text{CE}(T,\pi_\theta)}$	mam: $\frac{\text{CE}(T,\pi_\theta)}{H(p_{true})}$	mam: $\frac{\text{mtf_frq}(\pi_\theta)}{\text{mtf_frq}(r)}$
rs	500	1.24 ± 0.07	1.19 ± 0.07	$[32.0, 392.0]$	1.23 ± 0.03	1.16 ± 0.03	$[59.26, 433.33]$
rs	1000	1.24 ± 0.07	1.16 ± 0.07	$[23.87, 653.33]$	1.21 ± 0.03	1.14 ± 0.03	$[26.29, 233.33]$
rs	5000	1.18 ± 0.08	1.09 ± 0.05	$[3.59, 206.67]$	1.16 ± 0.05	1.08 ± 0.04	$[7.32, 130.0]$
rs	10000	1.08 ± 0.1	1.04 ± 0.02	$[0.89, 196.0]$	1.02 ± 0.03	1.04 ± 0.03	$[1.0, 4.97]$
rs	20000	0.99 ± 0.01	1.02 ± 0.01	$[0.81, 1.76]$	0.99 ± 0.0	1.02 ± 0.0	$[0.85, 1.04]$

Table 3: Overall statistics: for $D_m, motif \in \{10001010001, 01011101101, 001001100111, 1011100111001, 10001011111000\}, ft \in \{1001111, 1011111, 0111111\}$ and D_{mam}, $motif \in \{01011101101, 001001100111, 1011100111001, 100010100011, 10001011111000\}, ft \in \{1001111\}$.

motif, then the motif feature itself is not fully predictive. It can be seen in panel (d) of Figure 2 that the π_θ achieved with P_λ trained on mixture D_{mam} has consistent behaviour with the results obtained on the pure D_m of panels (a,b,c).

8.4 Regimes in Training-1

For training GAM we consider two methods, $snis$ and rs. As described in the previous sections, their impact on P_λ leads to π_θ's that have similar CE's and motif frequencies. Despite such resemblance in terms of accuracy, these two methods differ in terms of speed (see Table 2). Namely, when r is close to white noise due to small $|D|$, then for the rare events rs rejects most samples not containing the motif due to the effect of the log linear term and negative value of the component λ_m corresponding to the m feature, while $snis$ is able to exploit all samples. Despite being faster than rs, $snis$ remains competitive in terms of CE.

8.5 Cyclical vs two-stage training

We conducted a small experiment to compare the performance of cyclical training with two-stage training in terms of speed and accuracy for a fixed motif m and features ft (see [SM] Table 4, Figure 3). We observed that CEs of the obtained π_θ's were about the same for different values of $|D|$ and Training-1 regimes. On the other hand, there was no systematic improvement in the training speed of one method over the other.

9 Discussion

The basic idea behind GAMs is very simple. First, we extend the representational power of the autoregressive model r by multiplying by a log-linear potential, obtaining an unnormalized model P_λ (Training-1). Then we try to "project" this extended representation again to an autoregressive model π_θ (Training-2). Our results showed that, under favorable prior knowledge conditions, the final π_θ was able to perform as well, when trained on small data, as the standard r, trained on large data. During our experiments, we noticed that training P_λ was actually easier than training π_θ from it. Intuitively, the small number of parameters to be fitted in the log-linear model requires less work and fewer data than the training of an autoregressive component.[6]

It is interesting to relate our study to certain aspects of Reinforcement Learning (RL).

First, consider Training-2. There, we have a "score" P_λ that we are trying to approximate through an autoregressive model π_θ, which is basically a sequential "policy". The main difference with RL is that we are not trying to find a policy that *maximizes* the score (which would be a bad idea for language modelling, as it would tend to concentrate the mass on a few sequences), but one that approximates P_λ in a *distributional* sense; our current distillation technique is only one way to approach this problem, but other techniques more in the spirit of RL are possible, a direction that we leave for future work.

Second, consider Training-1. Our approach, consisting in suggesting to the model a number of prior features, might look too easy and suspicious. But notice that in RL, one would typically directly provide to the model an externally defined *reward*, a very strong form of prior knowledge. Here, instead, we "only" indicate to the models which features it might attend to, and Training-1 then determines the "reward" P_λ through max-likelihood, a milder form of prior knowledge, more respectful for what the data has to say.[7]

Acknowledgements

Thanks to Matthias Gallé and Ioan Calapodescu for comments on a previous version of this paper and to the anonymous reviewers for their detailed reading and feedback.

[6] At a deeper level, there are extreme situations where the P_λ obtained at the end of Training-1 can perfectly represent the true process, but where no autoregressive model can actually fit P_λ: one way to obtain such situations consists in generating binary strings that satisfy a certain cryptographic predicate, associated with a specific feature; the importance of this feature can be easily detected through Training-1, but an autoregressive model has no chance of generalizing from distilled or true data, even in large quantities.

[7] We could say that while Training-2 addresses a question directly related to Reinforcement Learning, Training-1 addresses one related to *Inverse* Reinforcement Learning (Russell, 1998; Ng and Russell, 2000): it derives a reward from training evidence rather than imposing it externally.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks.

Marc G. Bellemare, Will Dabney, and Rémi Munos. 2017. A Distributional Perspective on Reinforcement Learning. *arXiv:1707.06887 [cs, stat]*. ArXiv: 1707.06887.

Rafael C. Carrasco. 1997. Accurate computation of the relative entropy between stochastic regular grammars. *Theoretical Informatics and Applications*, 31:437–444.

Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. 2008. On the computation of the relative entropy of probabilistic automata. *Int. J. Found. Comput. Sci.*, 19(1):219–242.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *CoRR*. Cite arxiv:1705.03122.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Cong Duy Vu Hoang, Ioan Calapodescu, and Marc Dymetman. 2018. Moment Matching Training for Neural Machine Translation: A Preliminary Study.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Tony Jebara. 2013. Log-Linear Models, Logistic Regression and Conditional Random Fields.

Michael I. Jordan. 2010. Chapter 8 The exponential family : Basics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu Jie Huang. 2006. A Tutorial on Energy-Based Learning. *Predicting Structured Data*, pages 191–246.

Andrew Y. Ng and Stuart J. Russell. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Art Owen. 2017. Adaptive Importance Sampling (slides).

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Christian P. Robert and George Casella. 2005. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Stuart Russell. 1998. Learning agents for uncertain environments (extended abstract). In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 101–103, New York, NY, USA. ACM.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*, second edition. The MIT Press.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Y. Bengio and J. S. Senecal. 2008. Adaptive Importance Sampling to Accelerate Training of a Neural Probabilistic Language Model. *Ieee Transactions on Neural Networks*, 19(4):713–722.

Learning Analogy-Preserving Sentence Embeddings for Answer Selection

Aïssatou Diallo[†,‡] **, Markus Zopf**[‡] **and Johannes Fürnkranz**[‡]
[†]Research Training Group AIPHES
[‡]Knowledge Engineering Group
Department of Computer Science, Technische Universität Darmstadt
{diallo@aiphes, mzopf@ke, juffi@ke}.tu-darmstadt.de

Abstract

Answer selection aims at identifying the correct answer for a given question from a set of potentially correct answers. Contrary to previous works, which typically focus on the semantic similarity between a question and its answer, our hypothesis is that question-answer pairs are often in analogical relation to each other. Using analogical inference as our use case, we propose a framework and a neural network architecture for learning dedicated sentence embeddings that preserve analogical properties in the semantic space. We evaluate the proposed method on benchmark datasets for answer selection and demonstrate that our sentence embeddings indeed capture analogical properties better than conventional embeddings, and that analogy-based question answering outperforms a comparable similarity-based technique.

1 Introduction

Answer selection is the task of identifying the correct answer to a question from a pool of candidate answers. The standard methodology is to prefer answers that are semantically similar to the question. Often, this similarity is strengthened by bridging the lexical gap between the text pairs via learned semantic embeddings for words and sentences. The main drawback of this method is that question-answer (QA) pairs are modeled independently, and that the correspondence between different pairs is not considered in these embeddings. In fact, these methods only focus on the relationship that may exist between the entities that constitutes the QA pair at hand and are thus, limited to pairwise semantic structures.

Instead, we argue in this paper that questions and their correct answers often form analogical relations. For example, the question "Who is the president of the United States?" and its answer are

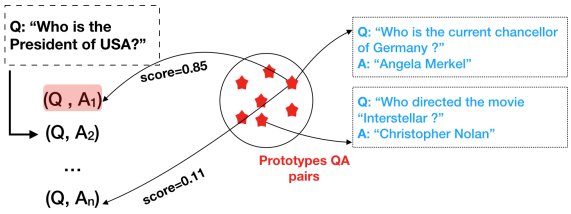


Figure 1: Illustration of analogy-based answer selection. Given a question and its candidate answers, each pair is compared to a QA prototype pair. The candidate answer with the highest score is assumed to be the correct answer.

in the same relation to each other as the question "Who is the current chancellor of Germany?" and "Angela Merkel". Thus, for modelling these relations, we need to look at quadruples of textual items in the form of two question-answer pairs, and want to reinforce that they are in the same relation to each other.

We expect that using analogies to identify and transfer positive relationships between QA pairs will be a better approach for tackling the task of answer selection than simply looking at the similarity between individual questions and their answers.

We use sentence embeddings as the mechanism to assess the relationship between two sentences, and aim to learn a latent representation in which their analogical relation is explicitly enforced in the latent space. Analogies are defined as relational similarities between two pairs of entities, such that the relation that holds between the entities of the first pair, also holds for the second pair. Loosely speaking, the quadruple of sentences is in *analogical proportion* if the difference between the first question and its answer is approximately the same as the difference between the second question and its answer.

This formulation is especially valuable because

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 910–919
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

analogies allow to put on relation pairs that are not directly or explicitly linked. Consequently, in the vector space, analogous QA pairs will be oriented in the same direction, whereas dissimilar pairs will not correspond.

The remainder of the paper is organized as follows: the next section will present related work on answer selection, metric learning, as well as laying down the foundations of analogical reasoning. In Section 3, we formally define analogies, and introduce our approach for learning such analogical embeddings. Finally, in Section 4, we evaluate the learnt representations to demonstrate that the found embeddings indeed respect the sought analogies, and to illustrate the benefits of analogies for the task of answer selection.

2 Related Work

Answer Selection. Answer selection is an important problem in natural language processing that has drawn a lot of attention in the research community (Lai et al., 2018). Given a question and a set of candidate answers, the task is to identify the correct answer(s) in this set. This task can be formulated as a classification or a ranking problem. Early works relied on computing a matching score between a question and its correct answer, and were characterized by the heavy reliance on feature engineering for representing the QA pairs. Representative works include (Filice et al., 2016), which studies the effects of various similarity, heuristic, and thread-based features, or (Tymoshenko and Moschitti, 2015), which analyzes the effect of syntactic and semantic features extracted by syntactic parser for answer re-ranking. Recently, deep learning methods have achieved excellent results in mitigating the difficulty of feature engineering. These methods are used to learn latent representations for questions and answers independently, and a matching function is applied to give the score of the two texts. The most representative works in this line of work include (Wang and Nyberg, 2015; Yin et al., 2016; Severyn and Moschitti, 2015; Tay et al., 2017).

Embeddings and Metric Learning. Our work is also related to representation learning usig deep neural networks. In fact, learning the embeddings of entities can be seen as a knowledge induction process, as those induced latent representations can be used to infer properties of unseen samples. Although many studies confirmed that embeddings obtained from distributional similarity can be useful in a variety of different tasks, (Levy et al., 2015) showed that the semantic knowledge encoded by general-purpose similarity embeddings is limited, and that enforcing the learnt representations to distinguish functional similarity from relatedness is beneficial. For this purpose, many task-specific embeddings have been proposed for a variety of tasks including (Riedel et al., 2013) for binary relation extraction and (FitzGerald et al., 2015) for semantic role labeling. This work aims to preserve more far reaching structures, namely analogies between pairs of entities.

Analogical Reasoning. Analogical reasoning has been an active research topic in classic artificial intelligence. It has been successfully used in different domains such as classification (Bounhas et al., 2014), clustering (Marx et al., 2002), dimensionality reduction (Memisevic and Hinton, 2004), or learning to rank (Fahandar and Hüllermeier, 2018). Gentner (1983) studies analogies with respect to human cognition, defines an analogy as a relational similarity over two pairs of entities, and differentiates it from the more superficial similarity defined by attributes. Since this general definition of analogy requires high-level reasoning which is not scalable to large-scale automated prediction systems, Miclet et al. (2008) define the concept of analogical dissimilarity between entities in the same semantic universe. The analogical dissimilarity allows to perform direct inference for unseen entities. Contrary to their direct inference setting, we enforce the analogical constraints in the learned embedding in the form of geometrical constraints, by imposing the co-linearity of the vector that maps the entities of each pair in the analogical proportion. It is worth mentioning that analogies have been found as the result of several word embedding models—inter alia (Mikolov et al., 2013; Pennington et al., 2014)—but those are allegedly only empirical observations, which we found to not carry over to our task.

3 Analogical Embeddings

In this section, we explain our approach towards generating semantic embeddings that preserve analogical proportions.

3.1 Analogical Reasoning

In this section, we briefly introduce key concepts in analogical reasoning, starting with analogical proportions.

Definition 1 (Analogical Proportion) *Let a, b, c, d be four values from a domain $\mathbb{X}$. The quadruple (a, b, c, d) is said to be in analogical proportion $a : b :: c : d$ if a is related to b as c is related to d, i.e., $\mathcal{R}(a, b) \sim \mathcal{R}(c, d)$.*

This comparative relation between two pairs of entities can be expressed in many ways (Dubois et al., 2016), but the most noteworthy are:

- Arithmetic proportion: $(a - b) = (c - d)$
- Geometric proportion: $\frac{\min(ad, bc)}{\max(ad, bc)}$

In this work, we focus solely on the arithmetic interpretation of analogy.

An intuitive way of viewing analogies is through geometrical constraints in an Euclidean space. Enforcing the relational similarity between pairs of elements is equivalent to constraining the four elements to form a parallelogram.

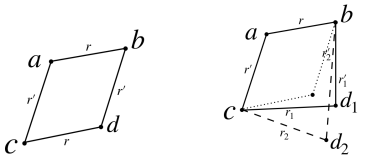
The left graph of Figure 2 illustrates such an *analogical parallelogram*. As we can see, in an *analogical parallelogram*, there is not only a relation $\mathcal{R}$ holding between (a, b) and (c, d) respectively, but there must also hold a similar relation $\mathcal{R}'$ between (a, c) and (b, d).

We can now make a first step towards our problem, which is learning to identify correct answers according using analogical inference. Given the aforesaid quadruple, when one of the four elements is unknown, an analogical proportion becomes an analogical equation.

Definition 2 (Analogical Equation) *An* analogical equation *has the form*

$$a : b :: c : x \tag{1}$$

where x represents an unknown element that is in analogical proportion to a, b, c.



(a) (b)

Figure 2: Analogical parallelograms in $\mathbb{R}^n$. (a) shows the case where $(a - b) = (c - d)$. The geometrical structure is a parallelogram. If $(a - b) \sim (c - d)$, the resulting structure is a general quadrangle with almost parallel sides (b).

In our setting, an exact solution to an analogical equation can often not be expected. Instead, we aim at finding the element d_i, among n candidates, where the analogical proportion is as closely satisfied as possible. For example, in the right graph of Figure 2, neither d_1 nor d_2 are perfect solutions to the analogical equation $a : b :: c : x$, but d_1 seems to be a better solution than d_2.

In order to relax the equality constraint between the pairs of entities, and to generalize the formulation of analogical proportions beyond the Boolean case, Miclet et al. (2008) proposed to measure the degree of an analogical proportion using analogical dissimilarity.

Definition 3 (Analogical Dissimilarity) *In a Euclidean space, the degree of analogical dissimilarity of a quadruple (a, b, c, d) is defined as*

$$v(a, b, c, d) = \|(a - b) - (c - d)\| \tag{2}$$

This equation represents the relation $\mathcal{R}$ as the difference between the entities of the pair and $\sim$ as the difference between the previously the so expressed relation pairs. Obviously, $v(a, b, c, d) = 0$ if (a, b, c, d) are in analogical proportion, and the value increases the less similar $(a - b)$ and $(c - d)$ are to each other.

This allows us to re-frame the original problem of answer selection as a ranking problem, in which the goal is to select the candidate answer d which minimizes the degree of analogical dissimilarity:

$$d = \arg\min_i v(a, b, c, d_i)_{i=1,\ldots,N} \tag{3}$$

In the following sections, we will describe the details of the model by motivating the architectural choices.

3.2 Generating Quadruples

In this work, we consider QA pairs as relational data. We aim to transfer knowledge from pairs whose relation is well known, which we call *prototypes*, to unseen pairs. For this, we train a model to encode analogies in the latent representations of the sentences. For creating a instances of quadruples to train the model, we adapt state-of-the-art datasets.

An analogy quadruple has the following form:

$$[q_p : a_p :: q_i : a_{ij}]$$

"Where" questions	
`Sentence A`	"Where was Abraham Lincoln born?"
`Sentence B`	"On February 12, 1809, Abraham Lincoln was born Hardin County, Kentucky"
`Sentence C`	"Where was Franz Kafka born?"
`Sentence D`	"Franz Kafka was born on July 3, 1883 in Prague, Bohemia, now the Czech Republic."
"Who" questions	
`Sentence A`	"Who made the rotary engine automobile?"
`Sentence B`	"Mazda continued work on developing the Wankel rotary engine."
`Sentence C`	"Who discovered prions?"
`Sentence D`	"Prusiner won Nobel prize last year for discovering prions"
"When" questions	
`Sentence A`	"When was Leonardo da Vinci born?"
`Sentence B`	"Leonardo da Vinci was actually born on 15 April 1452 [...] "
`Sentence C`	"When did Mt St Helen last have significant eruption?"
`Sentence D`	"Pinatubo's last eruption [...] as Mt St Helen's did when it erupted in 1980."

Table 1: Example of analogy between sentences. `Sentence A` and `Sentence B` constitutes the prototype QA pair in the analogical quadruples `Sentence C` and `Sentence D` are the QA pair at hand.

where the q_p and a_p, respectively stand for the question and the answer of the prototype pair, whereas q_i is the i-th question and a_{ij} is the j-th candidate answer to q_i.

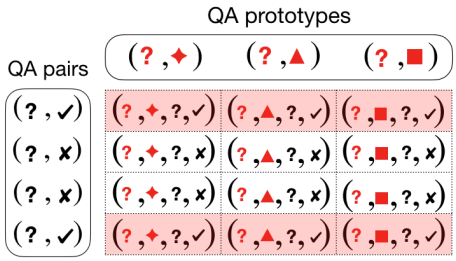


Figure 3: Procedure to generate analogical quadruples. The cells in red represent positive analogical quadruples, composed of a prototype QA pair, a question and its correct answer. In reverse, a negative quadruple contains a QA prototype, a question and one of its incorrect answer.

Given a set of questions and their relative candidates answers, we construct the analogical quadruples in two steps. First, we divide all the questions into three different subsets of wh-word questions: *"Who"*, *"When"* and *"Where"*. We focus on these three types because their answer type fall in distinct and easily identifiable categories:

- "Where" corresponds to an answer of type "Location"
- "Who" corresponds to an answer of type "Person"
- "When" corresponds to an answer of type "Date" or "Time"

Table 1 illustrates examples of quadruples for the three described categories. From these categories, we extract a variable number of QA pairs in order to form the prototype set. To generate positive quadruples, we select a prototype from one of the above-mentioned subsets and we associate a question from the same set and the correct answer among its candidates. This procedure provides a large number of analogical quadruples. On the other hand, to generate negative training samples we use the following approach: in the same subset, we associate a prototype, a question and a randomly selected wrong answer among its candidates. This is done in order to purposely break the analogical relation between a prototype QA pair and the QA pair at hand. This approach will generate a set of hard examples to help improve the training. Figure 3 illustrates the procedure.

To summarise, ranking by analogical dissimilarity is performed in three steps:

1. Given a prototype QA pair, a question and N candidate answers, N quadruples are generated.
2. The analogical dissimilarity score is computed for each quadruple.
3. The N candidates are consequently ranked by the analogical dissimilarity score.

The next section closely describe the architectural choices of the model.

3.3 Quadruple Siamese Network

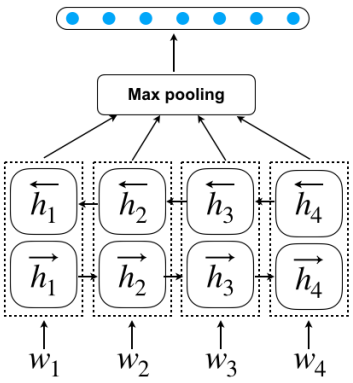


Figure 4: BiGRU with max pooling.

We recall that our focus is on learning an embedding function that pushes analogous QA pairs with similar mappings to be mutually close by enforcing a geometrical constraint in the vector space. This constraint states that the vector shift that maps the entities of the first pair should be similar to the vector shift of the second pair, according to the degree of analogical dissimilarity that holds between the two pairs $(a, b), (c, d)$.

To tackle this problem, we propose a Siamese network architecture as shown in Figure 5. In the next paragraphs, we describe the notation used and the details of each component of the model.

Notation. Let $\mathcal{Q}$ and $\mathcal{A}$ be the space of all questions and candidate answers. We denote a quadruple of sentences as (a, b, c, d), where $a, c \in \mathcal{Q}$ and $b, d \in \mathcal{A}$. Quadruples are assigned a label $y = 1$ if the analogical proportion holds, and 0 otherwise. θ denotes the parameters to be learnt that map the relation from a to b, and c to d respectively. Let $x.$ refer to the latent representation of one sentence in the quadruple.

Architecture. The Siamese network takes as input four sentences. The sub-networks in the Siamese model share the parameters and learn the vector representations for every sentence received as input. A sentence $S_i = w_{i1}, ...w_{ik}$ where w_{ij} represents the j^{th} word in the sentence S_i, $\forall i \in 1 \leq i \leq n$ and $\forall j \in 1 \leq j \leq k$. Words are mapped into word embeddings $x_{ij} = Ew_{ij}$, where $E^{d,|V|}$ is a matrix of vectors of size d, and V is the vocabulary. Out-of-vocabulary words are initialized by a random vector. We use bidirectional gated recurrent units (GRUs) (Cho et al., 2014) over the input sentence. For a sentence of T words, the network encodes T hidden states $h_1, ..., h_T$ such that:

$$\overrightarrow{h_t} = \overrightarrow{GRU}_t(w_1, ..., w_T)$$
$$\overleftarrow{h_t} = \overleftarrow{GRU}_t(w_1, ..., w_T)$$
$$h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$$

In order to obtain a fixed-size vector, we select the maximum value over each dimension of h_t using max pooling. After this step, we obtain four vectors of dimension d, one for each input sentence of the quadruple.

Training Strategy. The next step is to get the semantic relation between the pairs of input sentences. Given a pair a vectors, (x_i, x_j), the arithmetic proportion expects the difference of the vectors to encode the relational similarity between the entities that constitutes the pair. We let the network predict four d-dimensional embedding vectors, which we merge through a pairwise subtraction. Let $f_W(\cdot)$ be the projection of an input sentence in the embedding space computed by the network function f_W. Furthermore, let

$$f_{ab} = f_W(a) - f_W(b) \quad (4)$$
$$f_{cd} = f_W(c) - f_W(d) \quad (5)$$

be the pairwise differences between the embedding vectors. In order to separate instances of analogical proportion, similar pairs need to be mapped mutually close to each other, whereas dissimilar instances should be pushed apart.

For the energy of the model, we use the cosine similarity between the vector shifts of each pair of the quadruple:

$$E_W(f_{ab}, f_{cd}) = \frac{f_{ab} \cdot f_{cd}}{\|f_{ab}\|\|f_{cd}\|} \quad (6)$$

We argue that this is an appropriate energy function since the goal is for the pairs of parallel vectors to be parallel which maximises the analogical parallelogram likelihood.

We propose to use the contrastive loss (Hadsell et al., 2006) to perform the metric learning. This loss function has two terms, one for the similar and and another dissimilar samples. The similar instances are denoted by a label $y = 1$ whereas the dissimilar pairs are represented by $y = 0$. Thus, the loss function has the following form:

$$\mathcal{L}_W = y\,\mathcal{L}_+(f_{ab}, f_{cd}) + (1-y)\mathcal{L}_-(f_{ab}, f_{cd}) \quad (7)$$

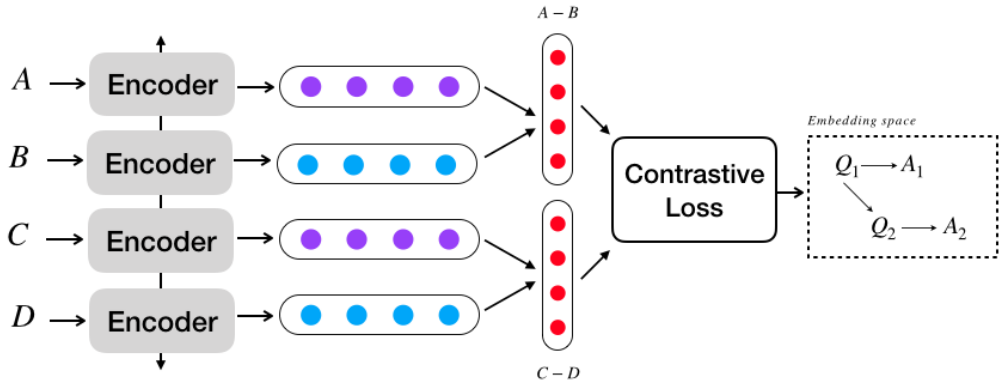


Figure 5: Siamese architecture.

Each term is expressed by:

$$\mathcal{L}_+(f_{ab}, f_{cd}) = (1 - E_W)^2 \quad (8)$$

$$\mathcal{L}_-(f_{ab}, f_{cd}) = \max((E_W - m)^2, 0) \quad (9)$$

This loss function measures how well the model learns to encode similar transformations such that analogous pairs are mutually close and form an analogical parallelogram in the embedding space, while pushing dissimilar transformations apart. Given a question and a pool of candidate answers, the goal is to rank the correct answer in the first position, based on how well each sentence completes the analogical equation according to (6).

This architecture is summarized in Figure 5. We learn all the parameters of the model through a gradient based method that minimizes the L2-regularized loss. Further details about the implementation are given in section 4.1.

4 Experiment

In this section, we present an evaluation of our approach in two experiments: first, in Section 4.2, we confirm that the found analogical embeddings do indeed improve the analogical parallelogram structure illustrated in Figure 2 over commonly used word- and sentence-based embeddings. In Section 4.3 we then show that this also results in improved performance for question answering. Before that, we start with a brief description of our experimental setup.

4.1 Experimental Setup

We begin the assessment of our model with a direct evaluation, which is ranking candidate answers in the same setting as during the training of the embedding. We generate quadruples with the same prototypes used for the training and we look for the correct answers by iteratively solving the analogical equations. We compare our model to commonly used sentence representations methods to evaluate the proposed approach results with respect to general purpose sentence embedding and word embedding methods. In the next paragraphs we present the experimental setup and the results obtained.

Datasets. We validate the proposed method on two datasets: WikiQA (Yang et al., 2015), an open domain QA dataset with answers collected over Wikipedia and TrecQA, which was created from the TREC Question Answer Track. Both resources are well established for benchmarking answer selection. We split each dataset into three subsets, which contain only "who", "where" and "when" questions. Table 2 reports the statistics of the two datasets.

	WikiQA			TrecQA		
type	train	dev	test	train	dev	test
"Who"	119	15	34	190	11	8
"When"	86	11	16	116	13	19
"Where"	71	17	22	96	9	11
Comb.	276	43	72	402	33	38

Table 2: Dataset by question type.

Evaluation metrics. We assess the performance of our method by measuring the Mean Average Precision (MAP) and the Mean Reciprocal Rank (MRR) for the generated quadruples in the test set. Given a set of questions Q, MRR is computed as follows:

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \quad (10)$$

where $rank_i$ represents the rank position of the first correct candidate answer for the i^{th} question. In other words, MRR is the average of the reciprocal ranks of results for the questions in set Q.

MAP is calculated as follows:

$$\text{MAP} = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{|m_j|}\text{Precision}(\pi_{jk}) \quad (11)$$

where $q_j \in Q$ is a question whose candidate answers are $a_1, ..., a_{m_j}$ and π_{jk} is the rank associated with those candidate answers. While MRR measures the rank of any correct answer, MAP computes the rank of all correct answers. Generally, MRR is higher than MAP on the same set of ranked objects.

Implementation details. We initiate the embedding layer with FastText vectors. These weights are not updated during training. The dimension of the output of the sentence encoder is 300. For alleviating overfitting we apply a dropout rate of 0.5. The model is trained with Adam optimizer with a learning rate of 0.001 and a weight decay rate of 0.01.

4.2 Quality of Analogical Embedding

Baselines. To support our claim that the learnt representations of our model encode the semantic of question answer pairs better than pre-trained sentence representation models, we choose four baselines commonly used to encode sentences:

1. *Word2Vec and Glove* (Mikolov et al., 2013; Pennington et al., 2014): We use the simple approach of averaging the word vectors for all words in a sentence. This method has the drawback of ignoring the order of the words of the sentence, but has shown to perform reasonably well.

2. *InferSent* (Conneau et al., 2017): Sentence embeddings obtained from training on Natural Language Inference dataset.

3. *Sent2Vec* (Pagliardini et al., 2017): A method to learn sentence embeddings such that the average of all words and n-grams can serve as sentence vector.

For each document in test set, we generate analogical quadruples as explained in section 3.2. Given a question q_i in the test set with k candidate answers, we obtain $p \times k$ possible quadruples, where p is the cardinality of the prototype set. The network encodes each sentence in the quadruple and computes the cosine similarity (6) between the obtained vector shifts.

Not every prototype QA pair will fit to the QA pair at hand, so we compute $p \times m$ scores, and choose only the prototype that leads to the highest analogical score for each document and discard the other comparisons. One might think about using the average of the scores and sorting the candidate answers accordingly, but this strategy introduces noise in the analogical inference procedure.

Results. We applied the described procedure to vectors obtained from our network as well as from the baseline representation methods. The results are shown in Table 3.

In order to better perceive the analogical properties of the baselines and the proposed approach, we also include a random baseline in the comparison. We observe that averaging word embeddings such as Glove or Word2Vec performs better than the dedicated sentence representations in the WikiQA dataset. This might be due to the fact that word embeddings have shown to encode some analogical properties. On the other hand, sentence embeddings have been trained with a particular learning objective, for example, InferSent has been train for the task of claim entailment with a classification objective and might not be suitable for representing relations between pairs of sentences. Nevertheless, ranking by the cosine similarity of the difference vectors do not lead to acceptable performances. This confirms our hypothesis that pre-trained sentence representation do not preserve analogical properties.

Similarly, we measure the influence of the number of prototypes on the performances.

	Model	WikiQA	
		MAP	MRR
W.E	Glove	0.464	0.475
	Word2Vec	0.4329	0.453
S.E	InferSent	0.399	0.404
	Sent2Vec	0.481	0.486
	This work	**0.6771**	**0.6841**

Table 3: Evaluation on quadruples. W.E. indicates averaging over word embeddings approach. S.E indicates dedicated sentence embeddings.

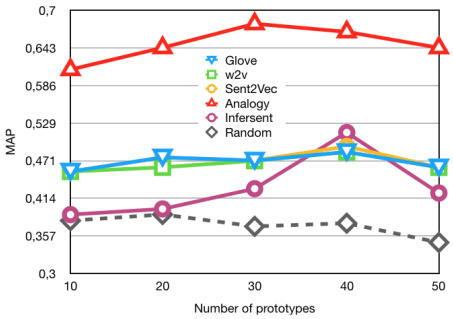


Figure 6: MAP for different number of prototypes.

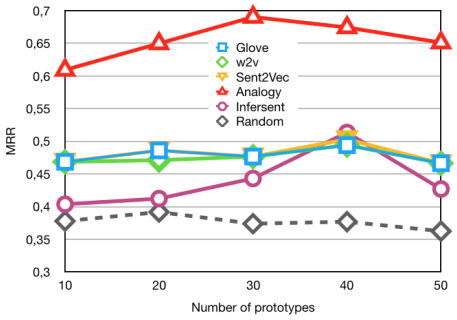


Figure 7: MRR for different number of prototypes.

We vary the number of prototypes pair $p \in \{10, 20, 30, 40, 50\}$ and measure the MAP and the MRR for both datasets. The results are shown in Figures 6 and 7. We can observe that the best performances are obtained for $p = 30$ and that after both MAP and MRR decrease. The reason might be that a high number of prototypes brings more comparisons and increases the probability of spurious interactions between QA prototypes and QA pairs.

4.3 Question-Answering Performance

A natural benchmark model for our work is the approach of Tam et al. (2017), which is similar to ours in that it proposed to replace wh-word in questions with appropriate named entities. This approach leverages typological information from a named entity recognizer and the word vector space.

It showed that simply replacing the wh-word, with a named entity that has the highest cosine similarity with all the candidate answers for a given question. This substitution is operated for ”where”, ”when” and ”who” types of questions. Finally, the transformed QA pair is fed to a network suited for the task of answer selection. This study demonstrated that this simple pre-processing step improves the state of the art results for the task of answer selection.

Alike our experimental setup, they divide the dataset in three categories, namely ”when”, ”who” and ”where”, which is the same division we used for our experimental setup, and evaluate their method on the split dataset and the full dataset. We will consider their work as our baseline in order to evaluate the capabilities of the analogy based embeddings. Moreover, we compare our approach to a setup which doesn't exploit analogical properties. This is to say, a Siamese network that takes as input a question and a candidate answer, generate the respective representations and compute the cosine similarity of the obtained sentence embeddings. The described baseline corresponds to the model proposed by (Tan et al., 2015) except for the fact that we use BiGRU for fair comparison.

	WikiQA		
	Baseline	Tam et al.	Analogy
”Who”	0.663	0.702	**0.763**
”When”	0.582	0.664	**0.701**
”Where”	0.568	**0.616**	0.602
Comb.	0.609	0.678	**0.684**

Table 4: MRR on WikiQA.

	TrecQA		
	Baseline	Tam et al.	Analogy
”Who”	0.787	0.781	**0.981**
”When”	0.797	**0.921**	0.863
”Where”	0.894	0.864	**0.929**
Comb.	0.837	0.875	**0.909**

Table 5: MRR on TrecQA.

The results are shown in Tables 4 and 5.

We observe that simply computing the cosine similarity between the difference vector of the prototype QA pair and the QA pair at hand with the learnt embedding from the proposed approach lead to significant improvements for some particular type of questions. The bold numbers in Tables 4 and 5 indicate the best results for each dataset. We can see that our method improves the MRR of at least two of questions types by a relevant margin. The last row of the same tables confirms that enforcing analogical properties in the embedding space generally improves the overall MRR for these three subsets.

5 Conclusion

This work introduced a new approach to learn sentence representations for answer selection, which preserve structural similarities in the form of analogies. Analogies can be seen as a way of injecting reasoning ability, and we express this by requiring common dissimilarities implied by analogies to be reflected in the learned feature space. We showed that explicitly constraining structural analogies in the learnt embeddings leads to better results over the distance-only embeddings. We believe that it is worth-while to further explore the potential of analogical reasoning beyond their common use in word embeddings, as it is a natural mean of learning and generalizing about relations between entities. The focus of this work has been on answer selection, but analogical reasoning can be useful in many other machine learning tasks such as machine translation or visual question answering. As a next step, we plan to explore other forms of analogies that involve modelling across domains.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1.

References

Myriam Bounhas, Henri Prade, and Gilles Richard. 2014. Analogical classification: A new way to deal with examples. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI)*. pages 135–140.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST@EMNLP)*. pages 103–111.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 670–680.

Didier Dubois, Henri Prade, and Gilles Richard. 2016. Multiple-valued extensions of analogical proportions. *Fuzzy Sets and Systems* 292:193–202.

Mohsen Ahmadi Fahandar and Eyke Hüllermeier. 2018. Learning to rank based on analogical reasoning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*. pages 2951–2958.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. Kelp at semeval-2016 task 3: Learning semantic relations between questions and answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT)*. pages 1116–1123.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 960–970.

Dedre Gentner. 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive Science* 7(2):155–170.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, volume 2, pages 1735–1742.

Tuan Manh Lai, Trung Bui, and Sheng Li. 2018. A review on deep learning techniques applied to answer selection. In *Proceedings of the 27th International Conference on Computational Linguistics*. pages 2132–2144.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. pages 970–976.

Zvika Marx, Ido Dagan, Joachim M. Buhmann, and Eli Shamir. 2002. Coupled clustering: A method for detecting structural correspondence. *Journal of Machine Learning Research* 3:747–780.

Roland Memisevic and Geoffrey E. Hinton. 2004. Multiple relational embedding. In *Advances in Neural Information Processing Systems 17*. pages 913–920.

Laurent Miclet, Sabri Bayoudh, and Arnaud Delhay. 2008. Analogical dissimilarity: definition, algorithms and two experiments in machine learning. *Journal of Artificial Intelligence Research* 32:793–824.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*. pages 3111–3119.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507* .

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics: Human Language Technologies (NAACL-HLT)*. pages 74–84.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 373–382.

Wai Lok Tam, Namgi Han, Juan Ignacio Navarro-Horñiacek, and Yusuke Miyao. 2017. Finding prototypes of answers for improving answer sentence selection. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. pages 4103–4108.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108* .

Yi Tay, Minh C. Phan, Anh Tuan Luu, and Siu Cheung Hui. 2017. Learning to rank question answer pairs with holographic dual LSTM architecture. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 695–704.

Kateryna Tymoshenko and Alessandro Moschitti. 2015. Assessing the impact of syntactic and semantic structures for answer passages reranking. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM)*. pages 1451–1460.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL), Volume 2: Short Papers*. pages 707–712.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 2013–2018.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics* 4:259–272.

A Simple and Effective Method for Injecting Word-level Information into Character-aware Neural Language Models

Yukun Feng[1], **Hidetaka Kamigaito**[1], **Hiroya Takamura**[1,2] **and Manabu Okumura**[1]
[1]Tokyo Institute of Technology
[2]National Institute of Advanced Industrial Science and Technology (AIST)
{yukun@lr.,kamigaito@lr.,takamura@,oku@}pi.titech.ac.jp

Abstract

We propose a simple and effective method to inject word-level information into character-aware neural language models. Unlike previous approaches which usually inject word-level information at the input of a long short-term memory (LSTM) network, we inject it into the softmax function. The resultant model can be seen as a combination of character-aware language model and simple word-level language model. Our injection method can also be used together with previous methods. Through the experiments on 14 typologically diverse languages, we empirically show that our injection method, when used together with the previous methods, works better than the previous methods, including a gating mechanism, averaging, and concatenation of word vectors. We also provide a comprehensive comparison of these injection methods.

1 Introduction

Language modeling (LM) is an important task in the natural language processing field, with various applications such as speech recognition (Mikolov et al., 2010a), machine translation (Koehn, 2009) and summarization (Filippova et al., 2015). Recently, neural language models (NLMs) have shown a great success and are better than traditional count-based methods (Bengio et al., 2003; Mikolov et al., 2010b). Standard NLMs usually maintain a fixed vocabulary and map each word to a continuous representation. These word representations obtained through NLMs are usually close to each other in the induced vector space if they are semantically similar. However, there are two main problems of standard NLMs. One is that they cannot handle out-of-vocabulary words. These words are usually replaced with a special unknown symbol. Another problem is that these models are not effective for learning the relationships between words for infrequent words. For example, although words "husbandman" and "salesman" share the suffix "man" in their surface forms, standard NLMs cannot capture such information in obtaining the relationship between the two words. A common way to deal with these issues is to use character information of each word to calculate the word representation, and it is often referred to as character-aware NLMs (Ling et al., 2015; Kim et al., 2016; Vania and Lopez, 2017; Gerz et al., 2018). Our research focuses on utilizing advantages of both character-level information and word-level information in character-aware NLMs.

Previous work usually combines word-level information and character-level information at the input of LSTM layers through a gating mechanism, or averaging or concatenation of word vectors. Because these approaches generally target at the input vectors, the word-level information cannot be explicitly taken into account at the output layer for predicting the next word.

To deal with this problem, we propose an improved character-aware neural language model that takes into account the injected word-level information at the output layer. This model is strongly inspired by the success of n-gram language models. Our model can predict the next word using the embeddings of the words in the current n-gram window, in addition to the hidden state of the LSTM layer. Specifically, we also use a gate to control how much word-level information should be taken before injecting it into the softmax function. After that, we combine the gated word-level information with the output of LSTM. Lastly, we feed these mixed information to the softmax function for word prediction. In our method, we can also take into account the information of previous words when injecting word-level information into the softmax function.

Our injection method is simple and easy to im-

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 920–928
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

plement [1]. We found our method effective compared with several common previous methods on 14 datasets with typologically diverse languages. In addition, the improvements can be further obtained when our injection method is used together with the previous methods. We also conducted a comprehensive comparison of these injection methods. Finally, we set up several experiments to check the effects of infrequent words on our model, and we also compared our model with several previous work on 6 common language modeling datasets. Our results show that:

- Compared with the previous injection methods (i.e., the gating mechanism, averaging, addition, and concatenation of word vectors), our injection method performs best on the majority of languages.

- Our injection method works effectively even when used alone, and the combination of our injection method and the previous injection methods performs better than the previous injection methods.

- When injecting word-level information into character-aware NLMs, discarding rare words in the training data can help improve the performance.

2 Related Work

Many work have attempted to improve character-aware NLMs in recent years. For example, Assylbekov and Takhanov (2018) proposed several ways of reusing weights in character-aware NLMs. Gerz et al. (2018) achieved an improved result on 50 typologically diverse languages by injecting subword-level information into word vectors at the softmax. For a thorough review of past researches, readers are recommended to read the work by Vania and Lopez (2017), who performed a systematic comparison across different models based on different subword units (characters, character trigrams, BPE, etc.).

One direction related to our research is to inject word-level information into character-aware neural models. Aside from language modeling, Santos and Zadrozny (2014) and dos Santos and Guimarães (2015) first used a convolutional neural network (CNN) to encode characters and then concatenated these encoded character-level representations and word-level representations for part-of-speech tagging and named entity recognition. Luong and Manning (2016) introduced a character-word neural machine translation model that only consults character-level representations for rare words encoded with a deep LSTM.

As research efforts for language models, Kang et al. (2011) used a simple character-word NLM designed for Chinese. Miyamoto and Cho (2016) introduced a gate mechanism between word embeddings and character embeddings obtained from a bidirectional LSTM (BiLSTM) for English. Verwimp et al. (2017) directly concatenated word and character embeddings without other subnetworks to encode the characters for English and Dutch.

Although there are a number of research efforts for using both character-level and word-level information, they feed the two types of information only to LSTM, while our model also injects the word-level information into the softmax function. Previous work on this topic has usually been tested in a limited number of languages and lacks a comprehensive comparison of different injection methods. We will compare our method with the previous methods mentioned in this section on 14 typologically diverse languages.

3 Model Description

For language modeling, we basically use a LSTM network (Hochreiter and Schmidhuber, 1997). We denote the hidden state of LSTM for the t-th word w_t as $\mathbf{h}_t \in \mathbb{R}^d$, where d is the embedding size. We incorporate word-level information using the neural network shown in Figure 1. We describe the details in the following subsections.

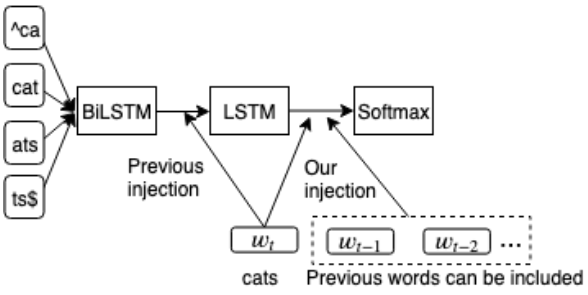


Figure 1: Our character-aware LSTM language model with injection of word-level information with an example word "cats". Symbols ^ and $ respectively represent the start and the end of a word.

[1] https://github.com/yukunfeng/char_word_lm

3.1 Input Word Representations

We use BiLSTM to encode character n-grams to obtain character-level representation. We set n to 3 for all the languages except Japanese and Chinese, for which we set n to 1. This is because BiLSTM over character 3-grams obtained best results on most LM datasets in the work of Vania and Lopez (2017), but Japanese and Chinese are more ideographic than the others, and it is expected that a smaller n works better.

Given a word w_t, we denote its embedding from a lookup table $\mathbf{W}_{in} \in \mathbb{R}^{d \times |V|}$ as $\mathbf{w_t} \in \mathbb{R}^d$, where $|V|$ is the vocabulary size. We compute the character-level representation of w_t as follows:

$$\mathbf{c}_t = \mathbf{W}_f \mathbf{h}_l^{fw} + \mathbf{W}_b \mathbf{h}_0^{bw} + \mathbf{b}, \quad (1)$$

where $\mathbf{h}_l^{fw}$, $\mathbf{h}_0^{bw} \in \mathbb{R}^d$ are the last states of the forward and backward LSTMs respectively. $\mathbf{W}_f$, $\mathbf{W}_b \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$ are trainable parameters. We define the following methods to obtain the combination $\mathbf{w}'_t$ from $\mathbf{w}_t$ and $\mathbf{c}_t$:

- **gate**: we use the same gating mechanism as Miyamoto and Cho (2016), which is described later to combine $\mathbf{w}_t$ and $\mathbf{c}_t$.
- **avg, add, cat**: we obtain $\mathbf{w}'_t$ through averaging, addition and concatenation of $\mathbf{w}_t$ and $\mathbf{c}_t$, respectively.

In the gating mechanism, we compute $\mathbf{w}'_t$ as follows:

$$g_{w_t}^{in} = \sigma\left(\mathbf{v}_g^\top \mathbf{w}_t + b_g\right), \quad (2)$$

$$\mathbf{w}'_t = (1 - g_{w_t}^{in})\mathbf{w}_t + g_{w_t}^{in}\mathbf{c}_t, \quad (3)$$

where $\mathbf{v}_g \in \mathbb{R}^d$ and $b_g \in \mathbb{R}$ are trainable parameters and $\sigma(\cdot)$ is a sigmoid function.

3.2 Representation of Input to Softmax

Our proposal is to combine $\mathbf{h}_t$ with $\mathbf{w_t}$ to better inform the softmax function of word-level information. Combination $\mathbf{h}'_t$ is computed as follows:

$$\mathbf{h}'_t = \mathbf{h}_t + g_{w_t}^{out}\mathbf{w}_t, \quad (4)$$

where $g_{w_t}^{out}$ is a gate value. In our experiments, we set up two types of gate. One is a fixed value, $g_{w_t}^{out} = 0.5$. The other is similar to the definition in Eq. (2), which adaptively outputs a gate value depending on w_t:

$$g_{w_t}^{out} = \sigma\left(\mathbf{v}_k^\top \mathbf{w}_t + b_k\right), \quad (5)$$

where $\mathbf{v}_k^\top \in \mathbb{R}^d$ and $b_k \in \mathbb{R}$ are trainable parameters. In Eq. (4), the gate is used only on word-level information to decide how much information $\mathbf{w}_t$ should be taken [2].

In Eq. (4), if we remove the term $\mathbf{h}_t$, the resultant model is a simple word-level language model $P(w_{t+1}|w_t)$. Based on this observation, we can simply extend our method to contain the word-level information for previous words without extra parameters:

$$\mathbf{h}_t^{word} = \sum_{i=1}^{n} \frac{1}{i}\mathbf{w}_{t+1-i}, \quad (6)$$

where n is the number of the current and previous words used to calculate $\mathbf{h}_t^{word}$. We simply give smaller weights inversely proportional to distance i to the embeddings of the previous words. For example, when $n = 2$, $\mathbf{h}_t^{word}$ is computed as $\mathbf{w}_t + \frac{1}{2}\mathbf{w}_{t-1}$, which is used to calculate $P(w_{t+1}|w_t, w_{t-1})$. The hidden state $\mathbf{h}'_t$ now can be calculated as follows:

$$\mathbf{h}'_t = \mathbf{h}_t + g_{w_t}^{out}\mathbf{h}_t^{word}. \quad (7)$$

3.3 Language Modeling

The language modeling task is to compute the probability of a given sentence $w_1, \ldots, w_T$:

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t|w_1, \ldots, w_{t-1}). \quad (8)$$

We use a softmax function based on $\mathbf{h}'_t$ to generate a probability distribution over the vocabulary:

$$P(w_{t+1}|w_1, \ldots, w_t) = \text{softmax}(\mathbf{W}_{out}^T \mathbf{h}_t{}'), \quad (9)$$

where $\mathbf{W}_{out} \in \mathbb{R}^{d \times |V|}$ is output word embeddings.

4 Model Variants

The hyper-parameters of our models are shown in Table 1. The learning rate is decreased if no improvement is observed in the validation dataset. Several baseline models and our models are listed as follows:

- **Char-BiLSTM-LSTM**: We use BiLSTM to encode characters without injecting word-level information.

[2] We have tested the above other methods, such as avg, add and cat, for combining $\mathbf{h}_t$ and $\mathbf{w}_t$, in place of gate, and found these methods did not work well.

Embedding size d	650
LSTM layers	2
Dropout	0.5
Optimizer	SGD
Learning rate	20
Learning rate decay	4
Parameter init: rand uniform	[-0.1,0.1]
Batch size	20
LSTM sequence length	35
Gradient clipping	0.25
Epochs	40

Table 1: Hyper-parameters of our model. We use d for the sizes of the character/word embeddings and for the number of hidden units of LSTM and BiLSTM.

- **Word-LSTM**: Standard word-level LSTM model.

- **Char-BiLSTM-gate/avg/add/cat-Word-LSTM**: We combine character-level and word-level information at the input of LSTM through gate/avg/add/cat methods, mentioned in Sec. 3.1.

- **Char-BiLSTM-LSTM-Word**: We inject word-level information only into the softmax function. This is our injection method.

- **Char-BiLSTM-gate/avg/add/cat-Word-LSTM-Word**: We combine our injection method and previous injection methods, which means we inject word-level information both at the input of LSTM and into the softmax function.

For both Char-BiLSTM-LSTM-Word and Char-BiLSTM-gate/avg/add/cat-Word-LSTM-Word, we use $g = 0.5$/adaptive and $n = 1/2/3$ to represent our specific injection method. For example, Char-BiLSTM-LSTM-Word ($g = 0.5, n = 2$) represents that we use a fixed gate value on word-level information in Eq. (4) and we inject the information of the current word and the preceding word into the softmax function.

5 Experiments on 14 Languages

5.1 Datasets

Common language modeling datasets for evaluating character-aware NLMs are from the work of Botha and Blunsom (2014). While these datasets contain languages with rich morphology, they have only 5 different languages. Perhaps, the most large-scale language modeling datasets are from the work of Gerz et al. (2018), who released 50 language modeling datasets covering typologically diverse languages. The difference between the newly released datasets and the previous common datasets is that unseen words are kept in test set. Thus, on the datasets, we can test our methods in a real LM setup. The languages from the work of Gerz et al. (2018) were selected to represent a wide spectrum of different morphological systems and contain many low-frequency or unseen words. Thus, these datasets should be desirable for checking the performance of character-aware NLMs [3].

To simplify the experiments without losing the wide coverage, we only chose datasets of 14 languages from these datasets and tried to cover different language typologies as well as different type/token ratios (TTRs). The statistics of our chosen datasets are shown in Table 2. We used all the words observed in training data and one special unknown token for out-of-vocabulary words as the output vocabulary to make the setting the same as Gerz et al. (2018).

5.2 Comparison of Baseline Models

The results of Word-LSTM and Char-BiLSTM-LSTM are shown in Table 3. We also showed the results of Word-LSTM and Char-CNN-LSTM from the work of Gerz et al. (2018). The embedding size and the number of LSTM layers are the same as those for the models in Gerz et al. (2018). As shown in the table, both the Word-LSTM and Char-BiLSTM-LSTM baselines are better than the Word-LSTM and Char-CNN-LSTM from the work of Gerz et al. (2018) on all the datasets [4]. Both Char-BiLSTM-LSTM and Char-CNN-LSTM from Gerz et al. (2018) are better than their respective Word-LSTM on all the datasets. One possible reason is that all the unseen words in the test set in the 14 datasets cannot be handled by Word-LSTM in the testing phase. However, character-aware models can encode the characters from these unseen words, making them possible to process these words. It is also shown that as TTR increases, Char-BiLSTM-LSTM achieves the better result than Word-LSTM. This may be because high-TTR languages have more low-frequency words and unseen tokens, as shown in Table 2. Since frequent

[3] To test our models against previous work, we also include experiments on common datasets, as described later.

[4] We have made the experimental setting the same as that of the work of Gerz et al. (2018), and the perplexity scores are comparable.

Language	Typology	TTR	Train vocab	#Train tokens	#Test tokens	#Unseen tokens	Freq<=15 (Train)
vi (Vietnamese)	Isolating	0.04	32055	754K	61.9K	1678	8.50%
zh (Chinese)	Isolating	0.06	43672	746K	56.8K	2132	16.00%
ja (Japanese)	Agglutinative	0.06	44863	729K	54.6K	2558	15.20%
pt (Portuguese)	Fusional	0.07	56167	780K	59.3K	2947	17.20%
en (English)	Fusional	0.07	55521	783K	59.5K	3618	16.60%
ms (Malay)	Isolating	0.07	49385	702K	54.1K	3918	16.00%
es (Spanish)	Fusional	0.08	60196	781K	57.2K	3486	17.90%
he (Hebrew)	Introflexive	0.12	83217	717K	54.6K	4855	27.20%
ar (Arabic)	Introflexive	0.12	89089	722K	54.7K	6076	26.40%
de (German)	Fusional	0.12	80741	682K	51.3K	5451	24.30%
cs (Czech)	Fusional	0.14	86783	641K	49.6K	5436	30.00%
ru (Russian)	Fusional	0.15	98097	666K	48.4K	4881	32.10%
et (Estonian)	Agglutinative	0.17	94184	556K	38.6K	4960	33.70%
fi (Finnish)	Agglutinative	0.20	115579	585K	44.8K	7899	38.10%

Table 2: The statistics of our language modeling datasets. TTR represents type/token ratio.

	vi	zh	ja	pt	en	ms	he	ar	de	cs	es	et	ru	fi
Word-LSTM (Gerz et al., 2018)	190	826	156	272	494	725	2189	2587	903	2200	366	2564	1309	4263
Char-CNN-LSTM (Gerz et al., 2018)	158	797	136	214	371	525	1519	1659	602	1252	275	1478	812	2236
Our Word-LSTM	137	582	113	201	348	476	1480	1610	609	1278	271	1295	839	2128
Char-BiLSTM-LSTM	134	578	107	178	302	463	1170	1337	483	973	230	967	620	1648
Char-BiLSTM-gate-Word-LSTM	136	582	112	195	328	483	1340	1619	551	1149	264	1189	704	1987
Char-BiLSTM-cat-Word-LSTM	133	565	105	183	314	432	1239	1360	504	1052	245	993	614	1602
Char-BiLSTM-avg-Word-LSTM	133	609	110	177	307	461	1181	1340	**478**	963	225	996	611	**1574**
Char-BiLSTM-add-Word-LSTM	127	551	103	171	298	423	1091	1302	481	938	218	967	606	1578
Char-BiLSTM-LSTM-Word (g =adaptive,$n = 1$)	126	567	104	175	314	424	1133	1279	491	**920**	235	949	605	1592
Char-BiLSTM-LSTM-Word ($g = 0.5, n = 1$)	**123**	**523**	**101**	**171**	**292**	**415**	**1068**	**1247**	479	934	**217**	**906**	**601**	1590

Table 3: Perplexity of several baseline models and Char-CNN-LSTM on 14 language modeling datasets. The best results among all models are in bold.

words still occupy the majority of both training and test data, injecting word-level information is still helpful for improving these character-aware models, as shown below.

5.3 Comparison of Different Injection Methods

The results of all the other different injection methods on 14 language modeling datasets are also shown in Table 3. In our experiments, Char-BiLSTM-gate-Word-LSTM underperforms Char-BiLSTM-LSTM on all the datasets. This indicates the gate method is not effective in our experiments. Char-BiLSTM-cat-Word-LSTM achieves better results than Char-BiLSTM-gate-Word-LSTM on all the datasets, but still underperforms Char-BiLSTM-LSTM on 8 out of 14 datasets. Char-BiLSTM-avg-Word-LSTM outperforms Char-BiLSTM-cat-Word-LSTM on 9 out of 14 datasets, which indicates the simple average method is better than the gating mechanism and the concatenation method in our tasks. However, Char-BiLSTM-avg-Word-LSTM still has no obvious improvements, compared with Char-BiLSTM-LSTM on most datasets.

We found some previous work also has similar results in the language modeling task. Kim et al. (2016) used a Char-CNN-LSTM model without injecting word-level information. They reported that some basic methods (e.g., concatenation, averaging and adaptive weighting schemes) for injecting word-level information degraded the performance of their Char-CNN-LSTM. Miyamoto and Cho (2016) showed the concatenation method for injecting word-level information into their Char-BiLSTM-LSTM also degraded their Word-LSTM model.

Char-BiLSTM-add-Word-LSTM achieves more improved results than Char-BiLSTM-LSTM on 13 out of 14 datasets and also performs best in general among Char-BiLSTM-avg/add/gate/cat-Word-LSTM. The addition method works better

than other previous injection methods in general in our tasks, while this simple method is less mentioned in the previous work. In conclusion, the performance of the previous injection methods in our experiments was in the descending order of add, avg, cat and gate.

Our Char-BiLSTM-LSTM-Word ($g = 0.5, n = 1$) and Char-BiLSTM-LSTM-Word ($g = \text{adaptive}, n = 1$) work effectively, and both of them achieve better results than Char-BiLSTM-LSTM. A simple fixed gate value in our injection method may be effective enough. Char-BiLSTM-LSTM-Word ($g = 0.5, n = 1$) works better than Char-BiLSTM-LSTM-Word ($g = \text{adaptive}, n = 1$) on most datasets. When compared with other injection methods, Char-BiLSTM-LSTM-Word ($g = 0.5, n = 1$) achieves the best results on most datasets (bold scores in Table 3). This suggests that our injection method, aiming at the different position from the input of LSTM, the softmax function, makes good use of word-level information.

5.4 Combination of Injection Methods

To avoid too many combinations of our injection method and other previous methods, we only chose to combine our Char-BiLSTM-LSTM-Word ($g = 0.5, n = 1$) with the other previous injection methods, because Char-BiLSTM-LSTM-Word ($g = 0.5, n = 1$) performs better than Char-BiLSTM-LSTM-Word ($g = \text{adaptive}, n = 1$), as mentioned above. The results of the combination of our Char-BiLSTM-LSTM-Word ($g = 0.5, n = 1$) and the previous injection methods are shown in Table 4.

When our injection method is used together with gate/avg/cat/add methods, obvious improvements can be observed on most datasets. Among them, Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1$) obtained the best results on most datasets (bold scores in Table 4). The result indicates that the previous injection methods do not make full use of word-level information, while our method, which injects the word-level information into the different position, specifically, the softmax, can help the previous models make better use of the word-level information.

5.5 Including Word-level Information for Previous Words

As mentioned in Sec. 3.1, we can include word-level information for previous words when injecting it into the softmax function. The number of words used in our injection method is denoted by n. In our experiments, we only set n to 1, 2 and 3, as we observed no obvious improvements when using a larger n. Since Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1$) performs best in general on most datasets, as mentioned above, we only changed n for this model. Note that our Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 2/3$) does not need extra parameters as we just reuse the word embeddings from the lookup table $\mathbf{W}_{in}$ to compute word-level information. In addition, the computational time of our injection method should be low, since the involved computation is simple. The result is shown in Table 5.

In general, Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 2$) achieves the best result on most datasets. Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 3$) does not obtain further improvements on most datasets. Since our current method for including word-level information for previous words is simple, a more advanced method can be further exploited in future work.

5.6 Effects of Infrequent Words

In order to check whether infrequent words help our character-aware NLMs, we set up several experiments by discarding some infrequent words based on their word frequency. Note that we maintain two independent vocabularies. One is the input vocabulary and is used to inject word-level information. We obtain the word embeddings in our and previous injection methods through the lookup table $\mathbf{W}_{in}$, as described in Sec. 3.1. The other is the output vocabulary and is used for word prediction, as described in Sec. 3.3. When we discard the infrequent words, we only narrow down the input vocabulary and do not change the output vocabulary. Thus, the perplexity scores are still comparable with the scores in the above experiments. For example, when our model processes the sentence "the salesman brought some samples" in training phase, where 'salesman' is an infrequent word in training data, our model can still try to predict the word 'salesman' given the previous word 'the', because 'salesman' is in our output vocabulary. When inputting the word 'salesman' to predict the word 'brought', we do not inject word-level information for the word 'salesman'. We only use its character-level representation obtained through our BiLSTM over characters to perform the lan-

	vi	zh	ja	pt	en	ms	he	ar	de	cs	es	et	ru	fi
Char-BiLSTM-gate-Word-LSTM	136	582	112	195	328	483	1340	1619	551	1149	264	1189	704	1987
Char-BiLSTM-gate-Word-LSTM-Word ($g = 0.5, n = 1$)	125	538	105	182	316	430	1339	1474	536	1116	260	1103	659	1728
Char-BiLSTM-cat-Word-LSTM	133	565	105	183	314	432	1239	1360	504	1052	245	993	614	1602
Char-BiLSTM-cat-Word-LSTM-Word ($g = 0.5, n = 1$)	122	541	103	180	305	426	1158	1316	530	1031	241	1012	607	1561
Char-BiLSTM-avg-Word-LSTM	133	609	110	177	307	461	1181	1340	478	963	225	996	611	1574
Char-BiLSTM-avg-Word-LSTM-Word ($g = 0.5, n = 1$)	121	495	99	165	293	398	1044	1224	488	890	218	898	569	1510
Char-BiLSTM-add-Word-LSTM	127	551	103	171	298	423	1091	1302	481	938	218	967	606	1578
Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1$)	**116**	**481**	**98**	**160**	**291**	**387**	**1038**	**1172**	**462**	**874**	**215**	**870**	**568**	**1494**

Table 4: Perplexity of the combination of our injection method with the previous methods on 14 language modeling datasets.

	vi	zh	ja	pt	en	ms	he	ar	de	cs	es	et	ru	fi
Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1$)	**116**	481	98	**160**	291	387	1038	1172	462	874	215	**870**	568	1494
Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 2$)	117	489	**95**	163	**277**	**376**	**998**	1179	**452**	**867**	**213**	884	**548**	**1456**
Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 3$)	118	**475**	96	162	285	391	1041	**1162**	463	877	215	913	563	1471

Table 5: Perplexity of our Char-BiLSTM-add-Word-LSTM-Word including word-level information for previous words on 14 language modeling datasets.

guage modeling task.

We denote the frequency threshold as θ and set its value among 5, 15 and 25. If the frequency of a word seen in the training data is less than or equal to θ, we discard it. We refer the model that discards infrequent words as Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1, \theta = 5/15/25$). The result is shown in Table 7.

When discarding the words whose frequency is less than or equal to 15, the model obtains better results only on 2 out of 14 datasets than Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1$). This indicates some infrequent words are still helpful. When we increase the frequency threshold further to 25, the performance of the model has dropped compared with Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1, \theta = 15$) as more frequent words are discarded. However, we found a relatively small frequency threshold $\theta = 5$ works quite effectively. Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1, \theta = 5$) achieves better results than Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1$) on 7 out of 14 datasets. It seems to be the trend that discarding infrequent words with $\theta = 5$ is useful for high TTR languages. Note that we arranged our datasets from low TTR to high TTR in Table 7. Since many of the words in natural languages are rare as described in Zipf's law, we can reduce the size of the input vocabulary significantly even with a small θ. The size for the full input vocabulary and the reduced vocabulary with different frequency threshold value is shown in Table 8. As we can see, when θ is set to 5, our model achieves better results with fewer parameters.

6 Experiments on 6 Common Datasets

In addition to the above datasets, we also set up 6 common language modeling datasets: English Penn Treebank (PTB) (Marcus et al., 1993) and 5 non-English datasets with rich morphology from the 2013 ACL Workshop on Machine Translation[5], which have been commonly used for evaluating character-aware NLMs (Botha and Blunsom, 2014; Kim et al., 2016; Bojanowski et al., 2017; Assylbekov and Takhanov, 2018). Since some of previous work has tested their model on PTB, we also included PTB in our experiment. We used the preprocessed small version of non-English datasets by Botha and Blunsom (2014) and followed the same split as the previous work. The data statistics is provided in Table 9.

The results of our proposed models and previous work are shown in Table 6. We used Char-BiLSTM-LSTM and Char-BiLSTM-add-Word-LSTM as baseline models. For our models, we set the frequency threshold θ to 5 and also set n to 2 as these settings help improve our character-aware NLMs, as discussed in Sec. 5.6 and Sec. 5.5. The language models used in the previous work are improved at different aspects, and most of them are also based on standard LSTM, like ours. Botha

[5]http://www.statmt.org/wmt13/translation-task.html

	PTB	CS	DE	ES	FR	RU
MLBL (Botha and Blunsom, 2014)	-	465	296	200	225	304
MorphSum (Kim et al., 2016)	-	398	263	177	196	271
CharCNN (Kim et al., 2016)	78.9	371	239	165	184	261
SkipGram initialization (Bojanowski et al., 2017)	-	312	206	145	159	206
MorphSum+RE+RW(Assylbekov and Takhanov, 2018)	**72.2**	338	222	157	172	210
Char-BiLSTM-LSTM	85.5	311	198	144	164	223
Char-BiLSTM-add-Word-LSTM	79.1	300	199	138	155	213
Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1, \theta = 5$)	75.9	287	**192**	**135**	152	**201**
Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 2, \theta = 5$)	76.1	**284**	193	137	**150**	202

Table 6: Perplexity of our models and previous work on 6 language modeling datasets.

	vi	zh	ja	pt	en	ms	he	ar	de	cs	es	et	ru	fi
Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1$)	**116**	**481**	**98**	**160**	291	**387**	1038	1172	**462**	874	215	**870**	568	1494
Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1, \theta = 5$)	**116**	495	**98**	166	**285**	397	**1016**	**1153**	463	**863**	**214**	877	**547**	**1492**
Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1, \theta = 15$)	117	502	99	164	286	397	1046	1185	467	883	215	924	570	1492
Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1, \theta = 25$)	118	502	101	167	292	405	1053	1202	471	896	215	929	573	1526

Table 7: Perplexity of Char-BiLSTM-add-Word-LSTM-Word ($g = 0.5, n = 1$) with different frequency thresholds on 14 language modeling datasets.

	Full	$\theta = 5$	$\theta = 15$	$\theta = 25$
vi	32055	5979	3383	2547
zh	43672	12200	5847	3940
ja	44863	9793	4355	2806
pt	56167	11207	4975	3203
en	55521	11142	5060	3282
ms	49385	9849	4728	3187
he	83217	14867	5961	3589
ar	89089	13459	5607	3482
de	80741	10290	4020	2511
cs	86783	12581	4680	2762
es	60196	11043	4722	2959
et	94184	10392	3815	2299
ru	98097	13337	4677	2734
fi	115579	11520	3930	2303

Table 8: The size of input vocabulary seen in the training data on 14 datasets with different frequency threshold.

	Vocab size	#Train token
PTB	10K	1M
Czech (CS)	46K	1M
German (DE)	37K	1M
Spanish (ES)	27K	1M
French (FR)	25K	1M
Russian (RU)	86K	1M

Table 9: The data statistics of our 6 language modeling datasets.

and Blunsom (2014) used the morphological log-bilinear (MLBL) model, which takes into account morpheme information. Kim et al. (2016) used CNN as their character encoder, and also trained an LSTM language model, where the input representation of a word is the sum of the morpheme embeddings of the word. Bojanowski et al. (2017) trained the word embeddings through skip-gram models with subword-level information, and used these word embeddings to initialize the lookup table of word embeddings of a word-level language model. Assylbekov and Takhanov (2018) focused on reusing embeddings and weights in a character-aware language model. The input of their model is also the sum of the morpheme embeddings of the word. As shown in the table, Char-BiLSTM-LSTM underperforms the previous work on PTB. One reason may be that we did not tune the hyper-parameters of our models on PTB. The hyper-parameters were simply kept the same in all the experiments on 20 datasets. As we can see, Char-BiLSTM-LSTM achieves better results than most previous work on non-English datasets. Our models also achieve the best results on non-English datasets.

7 Conclusion

In addition to combining character-level and word-level information at the input of LSTM,

which is a widely used combination manner, we proposed to also inject word-level information into the softmax function in a character-aware neural language model. We gave a detailed comparison with previous methods, and the result showed our proposal works effectively on typologically diverse languages. For future work, it would be interesting to see how our model works for other tasks such as text generation.

Acknowledgments

We would like to thank anonymous reviewers for their constructive comments.

References

Zhenisbek Assylbekov and Rustem Takhanov. 2018. Reusing weights in subword-aware neural language models. In *NAACL-HLT*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *International Conference on Machine Learning*, pages 1899–1907.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368.

Daniela Gerz, Ivan Vulić, Edoardo Ponti, Jason Naradowsky, Roi Reichart, and Anna Korhonen. 2018. Language modeling for morphologically rich languages: Character-aware modeling for word-level prediction. *Transactions of the Association of Computational Linguistics*, 6:451–465.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Moonyoung Kang, Tim Ng, and Long Nguyen. 2011. Mandarin word-character hybrid-input neural network language model. In *Twelfth Annual Conference of the International Speech Communication Association*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernández Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010a. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010b. Recurrent neural network based language model. In *INTERSPEECH*.

Yasumasa Miyamoto and Kyunghyun Cho. 2016. Gated word-character recurrent language model. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1992–1997. Association for Computational Linguistics.

Cicero dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of the Fifth Named Entity Workshop*, pages 25–33. Association for Computational Linguistics.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2016–2027, Vancouver, Canada. Association for Computational Linguistics.

Lyan Verwimp, Joris Pelemans, Hugo Van hamme, and Patrick Wambacq. 2017. Character-word lstm language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 417–427. Association for Computational Linguistics.

On Model Stability as a Function of Random Seed

Pranava Madhyastha
Department of Computing
Imperial College London
pranava@imperial.ac.uk

Rishabh Jain*
Bloomberg
London
rjain213@bloomberg.net

Abstract

In this paper, we focus on quantifying model stability as a function of random seed by investigating the effects of the induced randomness on model performance and the robustness of the model in general. We specifically perform a controlled study on the effect of random seeds on the behaviour of attention, gradient-based and surrogate model based (LIME) interpretations. Our analysis suggests that random seeds can adversely affect the consistency of models resulting in counterfactual interpretations. We propose a technique called *Aggressive Stochastic Weight Averaging (ASWA)* and an extension called *Norm-filtered Aggressive Stochastic Weight Averaging (NASWA)* which improves the stability of models over random seeds. With our ASWA and NASWA based optimization, we are able to improve the robustness of the original model, on average reducing the standard deviation of the model's performance by 72%.

1 Introduction

There has been a tremendous growth in deep neural network based models that achieve state-of-the-art performance. In fact, most recent end-to-end deep learning models have surpassed the performance of careful human feature-engineering based models in a variety of NLP tasks. However, deep neural network based models are often brittle to various sources of randomness in the training of the models. This could be attributed to several sources including, but not limited to, random parameter initialization, random sampling of examples during training and random dropping of neurons. It has been observed that these models have, more often, a set of *random seeds* that yield better results than others. This has also lead to research suggesting random seeds as an additional hyper-parameter for tuning (Bengio, 2012)[1]. One possible explanation for this behavior could be the existence of multiple local minima in the loss surface. This is especially problematic as the loss surfaces are generally non-convex and may have multiple saddle points making it difficult to achieve model stability.

if high crimes were any more generic it would have a universal product code instead of a title

$(\Pr(Y_{negative}) = 0.99)$

if high crimes were any more generic it would have a universal product code instead of a title

$(\Pr(Y_{negative}) = 0.98)$

Figure 1: Importance based on attention probabilities for two runs of the same model with **same parameters and same hyperparameters**, but with **two different random seeds** (color magnitudes: pink<magenta<red)

Recently the NLP community has witnessed a resurgence in interpreting and explaining deep neural network based models (Jain et al., 2019; Jain and Wallace, 2019; Alvarez-Melis and Jaakkola, 2017). Most of the interpretation based methods involve one of the following ways of interpreting models: a) sample oriented interpretations: where the interpretation is based on changes in the prediction score with either upweighting or perturbing samples (Jain et al., 2019; Jain and Wallace, 2019; Koh and Liang, 2017); b) interpretations based on feature attributions using attention or input perturbation or gradient-based measures; (Ghaeini et al., 2018; Feng et al., 2018; Bach et al., 2015); c) interpretations using surro-

*This work was conducted when the author was a student at Imperial College London.

[1] http://www.argmin.net/2018/02/26/nominal/

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 929–939
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

gate linear models (Ribeiro et al., 2016) – these methods can provide local interpretations based on input samples or features. However, the presence of inherent randomness makes it difficult to accurately interpret deep neural models among other forms of pathologies (Feng et al., 2018).

In this paper, we focus on the stability of deep neural models as a function of random-seed based effects. We are especially interested in investigating the hypothesis focusing on model stability: do neural network based models under different random seeds allow for similar interpretations of their decisions? We claim that for a given model which achieves a substantial performance for a task, the factors responsible for any decisions over a sample should be approximately consistent irrespective of the random seed. In Figure 1, we show an illustration of this question where we visualize the attention distributions of two CNN based binary classification models for sentiment analysis, trained with the same settings and hyper-parameters, but with *different seeds*. We observe that both models obtain the correct prediction with significantly high confidence. However, we note that both the models attend to completely different sets of words. This is problematic, especially when interpreting these models under the influence of such randomness. We observe that on average $40-60\%$ of the most important interpretable units are different across different random seeds for the same model. This phenomenon also leads us to the question on the exact nature of interpretability – are the interpretations specific to an instantiation of the model or are they general to a class of models?

We also provide a simple method that can, to a large extent, ameliorate this inherent random behaviour. In Section 3.1, we propose an aggressive stochastic weight averaging approach that helps in improving the stability of the models at almost zero performance loss while still making the model robust to random-seed based instability. We also propose an improvement to this model in Section 3.2 which further improves the stability of the neural models. Our proposals significantly improve the robustness of the model, on average by 72% relative to the original model and on Diabetes (MIMIC), a binary classification dataset, by 89% (relative improvement). All code for reproducing and replicating our experiments is released in our repository[2].

2 Measuring Model Stability

In this section, we describe methods that we use to measure model stability, specifically — prediction and interpretation stability.

2.1 Prediction Stability

We measure prediction stability using standard measures of the mean and the standard deviations corresponding to the accuracy of the classification based models on different datasets. We ensure that the models are run with exactly the same configurations and hyper-parameters but with different random seeds. This is a standard procedure that is used in the community to report the performance of the model.

2.2 Interpretation Stability

For a given task, we train a set of models only differing with random-seeds. For every given test sample, we obtain interpretations using different instantiations of the models. We define a model to be stable if we obtain similar interpretations regardless of different random-seed based instantiations. We use the following metrics to quantify stability:

a) **Relative Entropy quantification ($\mathcal{H}$):** Given two distributions over interpretations, for the same test case, from two different models, it measures the relative entropy between the two probability distributions. Note that, the higher the relative entropy the greater the dissimilarity between the two distributions.

$$\mathcal{H} = \sum_{i \in d} \mathrm{Pr}_1 \cdot \log \frac{\mathrm{Pr}_1}{\mathrm{Pr}_2}$$

where, Pr_1 and Pr_2 are two attention distributions of the same sample from two different runs of the model and d is the number of tokens in the sample. Given n differently seeded models, for each test instance, we calculate the relative entropy obtained from the corresponding averaged pairwise interpretation distributions.

b) **Jaccard Distance ($\mathcal{J}$):** It measures the dissimilarity between two sets. Here higher values of $\mathcal{J}$ indicate larger variances. We consider top-n tokens which have the highest attention for comparison. Note that, Jaccard distance is over sets of

[2] https://github.com/rishj97/ModelStability

word indices and do not take into account the attention probabilities explicitly. Jaccard distance is defined as:

$$\mathcal{J} = (1 - \frac{A \cap B}{A \cup B}) * 100\%$$

where, A and B are the sets of most relevant items. We specifically decided to use 'most' relevant (top-n items) as the tail of the distribution mostly consists of values close to 0.

Interpretation methods under study: In this paper we study interpretation stability using the following three interpretation methods:

1. *Attention based interpretation:* We focus on attention probabilities as the mode of interpretation and consider the model to be stable if different instantiations of the model leads to similar attention distributions. Our major focus in this paper is attention based interpretation. As we use Jain et al. (2019) as a testbed for our investigation, we focus heavily on attention. Also, as the attention layer has a linear relationship with the prediction, we consider attention to be more indicative of the model stability.

2. *Gradient-based feature importance:* Given a sample, we use the input gradients of the model corresponding to each of the word representations and compute the magnitude of the change as a local explanation. We refer the reader to Baehrens et al. (2010) for a good introduction to gradient-based interpretations. As all of our models are differentiable, we use this as an alternative method for interpretation. We follow the standard procedure as followed in Feng et al. (2018) and note that we do not follow Jain and Wallace (2019) and do not disconnect the computational graph at the attention module. We obtain probabilistic gradient scores by normalizing over the absolute values of gradient values.

3. *LIME based interpretation:* We use locally interpretable model-agnostic interpretations (Ribeiro et al., 2016) that learns a surrogate interpretable model locally around the predictions of the deep neural based model. We obtain LIME based interpretations for every instantiation of the models. We then use Jaccard Distance to measure the divergence.

We note that, we observe similar patterns across the three interpretation methods and the interpretations consistently differ with random seeds.

3 Reducing Model Instability with an Optimization Lens

We observe that different instantiations of the model can cause the model have different starts on the optimization surface. Further, stochastic sampling might result in different paths. Both of these factors can lead to different local minimas potentially leading to different solutions. With this observation as our background we propose two, closely related, methods to ameliorate divergence: Agressive Stochastic Weight Averaging and Norm-filtered Agressive Stochastic Weight Averaging. We describe these two in the following subsections.

3.1 Aggressive Stochastic Weight Averaging (ASWA)

Stochastic weight averaging (SWA) (Izmailov et al., 2018) works by averaging the weights of multiple points in the trajectory of gradient descent based optimizers. The algorithm typically uses modified learning rate schedules. SWA is itself based on the idea of maintaining a running average of weights in stochastic gradient descent based optimization techniques (Ruppert, 1988; Polyak and Juditsky, 1992). The principle idea in SWA is averaging the weights that are maximally distant helps stabilize the gradient descent based optimizer trajectory and improves generalization. Izmailov et al. (2018) use the analysis of Mandt et al. (2017) to illustrate the stability arguments where they show that, under certain convexity assumptions, SGD iterations can be visualized as sampling from a Gaussian distribution centred at the *minima* of the loss function. Samples from high-dimensional Gaussians are expected to be concentrated *on the surface of the ellipse* and not close to the *mean*. Averaging iterations is shown to stabilize the trajectory and further improve the width of the solutions to be closer to the *mean*.

In this paper, we focus on the stability of deep neural models as a function of random-seeds. Our proposal is based on SWA, but we extend it to the extremes and call it *Aggressive Stochastic Weight Averaging*. We assume that, for small batch size, the loss surface is locally convex. We further relax

the conditions for the optimizer and assume that the optimizer is based on some version of gradient descent — this means that our modification is valid even for other pseudo-first-order optimization algorithms including Adam (Kingma and Ba, 2014) and Adagrad (Duchi et al., 2011).

We note that, Izmailov et al. (2018) suggest using SWA usually after 'pre-'training the model (at least until 75% convergence) and followed by sampling weights at different steps either using large constant or cyclical learning rates. While, SWA is well defined for convex losses (Polyak and Juditsky, 1992), Izmailov et al. (2018) connect SWA to non-convex losses by suggesting that the loss surface is *approximately* convex after convergence. In our setup, we investigate the utility of averaging weights over every iteration (an iteration consists of one batch of the gradient descent). Algorithm 1 shows the implementation pseudo-code for SWA. We note that, unlike Izmailov et al. (2018), we average our weights at *each batch* update and assign the ASWA parameters to the model at the end of each epoch. That is, we replace the model's weights for the next epoch with the averaged weights.

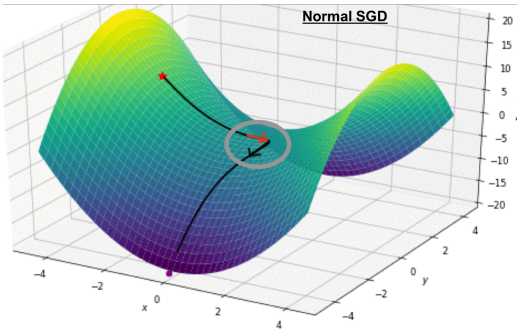
Algorithm 1: Aggressive SWA algorithm

Require:

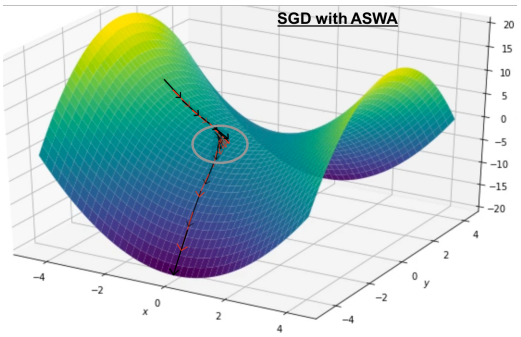
e = Epoch number
m = Total epochs
i = Iteration number
n = Total iterations
α = Learning rate
$\mathcal{O}$ = Stochastic Gradient optimizer function

$e \leftarrow 0$;
while $e < m$ **do**
  $i \leftarrow 1$
  while $i \leq n$ **do**
    $W_{swa} \leftarrow W_{swa} + \frac{(W - W_{swa})}{(e*n+i+1)}$;
    $W \leftarrow W - \mathcal{O}(\alpha, W)$;
    $i \leftarrow i + 1$
  $W \leftarrow W_{swa}$;
  $e \leftarrow e + 1$

In Figure 2, we show an SGD optimizer (with momentum) and the same optimizer *with SWA* over a 3-dimensional loss surface with a saddle point. We observe that the original SGD reaches the desired minima, however, it almost reaches the saddle point and does a course correction and reaches minima. On the other hand, we observe that SGD with ASWA is very conservative, it repeatedly restarts and reaches the minima without reaching the saddle point. We empirically observe that this is a desired property for the stability of models over runs of the same model that differ only over random instantiations. The grey circles in Figure 2 highlight this conservative behaviour of SGD with ASWA optimizer, especially when compared to the standard SGD. Further, Polyak and Juditsky (1992) show that for convex losses, averaging SGD proposals achieves the highest possible rate of convergence for a variety of first-order SGD based algorithms.



(a) Trajectory for Stochastic Gradient Descent



(b) Trajectory for Stochastic Gradient Descent with ASWA

Figure 2: Trajectory for gradient descent algorithms with red and black arrows on (b) indicating movements from consecutive epochs with restarts. Conservative behaviour of ASWA algorithm helps avoid the saddle point without ever reaching it.

3.2 Norm-filtered Aggressive Stochastic Weight Averaging (NASWA)

We observe that the ASWA algorithm is especially beneficial when the norm difference of the parameters of the model are high. We hypothesise that in general, the norm difference indicates the divergence between optimizers' steps and we observe that the larger the norm difference, the greater the change in the trajectory. Therefore, we propose to

Algorithm 2: Norm-filtered Aggressive SWA algorithm

Require:

e = Epoch number
m = Total epochs
i = Iteration number
n = Total iterations
α = Learning rate
$\mathcal{O}$ = Stochastic Gradient optimizer function
N_s = List of previous iterations' norm differences

$e \leftarrow 0$;
while $e < m$ **do**
  $i \leftarrow 1$
  while $i \leq n$ **do**
    $N_{cur} \leftarrow \|W - W_{swa}\|_1$;
    $N_{mean} \leftarrow \frac{\sum_{i=1}^{|N_s|} N_s[i]}{|N_s|}$;
    if $N_{cur} > N_{mean}$ **then**
      $W_{swa} \leftarrow W_{swa} + \frac{(W - W_{swa})}{(e*n+i+1)}$;
      $N_s \leftarrow [N_{cur}]$;
    else
      $N_s \leftarrow N_s + [N_{cur}]$;
    $W \leftarrow W - \mathcal{O}(\alpha, W)$;
    $i \leftarrow i + 1$
  $W \leftarrow W_{swa}$;
  $e \leftarrow e + 1$

maintain a list that stores the norm differences of the previous iterations. If the norm difference of the current iteration is greater than the average of the list, we update the ASWA weights and reinitialize the list with the current norm difference. When the norm difference, however, is less than the average of the list, we just append the current norm difference to the list. After the completion of the epoch, we assign the ASWA parameters to the model. This is shown in Algorithm 2. We call this approach *Norm-filtered Aggressive Stochastic Weight Averaging*.

4 Experiments

We base our investigation on similar sets of models as Jain and Wallace (2019). We also use the code provided by the authors for our empirical investigations for consistency and empirical validation. We describe our models and datasets used for the experiments below.

4.1 Models

We consider two sets of commonly used neural models for the tasks of binary classification and multi-class natural language inference. We use CNN and bi-directional LSTM based models with attention. We follow (Jain and Wallace, 2019) and use similar attention mechanisms using a) additive attention (Bahdanau et al., 2014); and b) scaled dot product based attention (Vaswani et al., 2017). We jointly optimize all the parameters for the model, unlike Jain and Wallace (2019) where the encoding layer, attention layer and the output prediction layer are all optimized separately. We experiment with several optimizers including Adam (Kingma and Ba, 2014), SGD and Adagrad (Duchi et al., 2011) but most results below are with Adam.

For our ASWA and NASWA based experiments, we use a constant learning rate for our optimizer. Other model-specific settings are kept the same as Jain and Wallace (2019) for consistency.

Dataset	Avg. Length	Train Size	Test size
IMDB	179	12500 / 12500	2184 / 2172
Diabetes(MIMIC)	1858	6381 / 1353	1295 / 319
SST	19	3034 / 3321	652/653
Anemia(MIMIC)	2188	1847 / 3251	460 / 802
AgNews	36	30000 / 30000	1900 / 1900
ADR Tweets	20	14446 / 1939	3636 / 487
SNLI	14	182764 / 183187 / 183416	3219 / 3237 / 3368

Table 1: Dataset characteristics. Train size and test size show the cardinality for each class. SNLI is a three-class dataset while the rest are binary classification

4.2 Datasets

The datasets used in our experiments are listed in Table 1 with summary statistics. We further pre-process and tokenize the datasets using the standard procedure and follow Jain and Wallace (2019). We note that IMDB (Maas et al., 2011), Diabetes(MIMIC) (Johnson et al., 2016), Anemia(MIMIC) (Johnson et al., 2016), AgNews (Zhang et al., 2015), ADR Tweets (Nikfarjam et al., 2015) and SST (Socher et al., 2013) are datasets for the binary classification setup. SNLI (Bowman et al., 2015) is a dataset for the multiclass classification setup. All of the datasets are in English, however we expect the behavior to persist regardless of the language.

4.3 Settings and Hyperparameters

We use a 300-dimenstional embedding layer which is initialized with FastText (Joulin et al., 2016) based free-trained embeddings for both CNN and the bi-directional LSTM based models.

We use a 128-dimensional hidden layer for the bi-directional LSTM and a 32-dimensional filter with kernels of size $\{1, 3, 5, 7\}$ for CNN. For others, we maintain the model settings to resemble the models in Jain and Wallace (2019). We train all of our models for 20 Epochs with a constant batch size of 32. We use early stopping based on the validation set using task-specific metrics (Binary Classification: using `roc-auc`, Multiclass and question answering based dataset: using `accuracy`).

Dataset	CNN(%)	CNN+ASWA(%)	CNN+NASWA(%)
IMDB	89.8 (±0.79)	90.2 (±0.25)	90.1 (±0.29)
Diabetes	87.4 (±2.26)	85.9 (±0.25)	85.9 (±0.38)
SST	82.0 (±1.01)	82.5 (±0.39)	82.5 (±0.39)
Anemia	90.6 (±0.98)	91.9 (±0.20)	91.9 (±0.19)
AgNews	95.5 (±0.23)	96.0 (±0.11)	96.0 (±0.07)
Tweet	84.6 (±2.65)	84.4 (±0.54)	84.4 (±0.54)

Table 2: Performance statistics obtained from 10 differently seeded CNN based models. Table compares accuracy and its **standard deviation** for the normally trained CNN model against the ASWA and NASWA trained models, whose deviation drops significantly, thus, indicating increased robustness.

5 Results

In this section, we summarize our findings for 10 runs of the model with 10 different random seeds but with identical model settings.

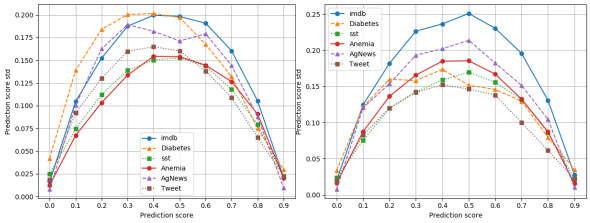
5.1 Model Performance and Stability

We first report model performance and prediction stability. The results are reported in Table 2.

Dataset	LSTM(%)	LSTM+ASWA(%)	LSTM+NASWA(%)
IMDB	89.1 (±1.34)	90.2 (±0.32)	90.3 (±0.17)
Diabetes	87.7 (±1.44)	87.7 (±0.60)	87.8 (±0.55)
SST	81.9 (±1.11)	82.0 (±0.60)	82.1 (±0.57)
Anemia	91.6 (±0.49)	91.8 (±0.34)	91.9 (±0.36)
AgNews	95.5 (±0.32)	96.1(±0.17)	96.1 (±0.10)
Tweet	84.7 (±1.79)	83.8 (±0.45)	83.9 (±0.45)

Table 3: Performance statistics obtained from 10 differently seeded LSTM based models.

We note that the original CNN based models, on an average, have a standard deviation of $\pm 1.5\%$. Which seems standard, however, we note that ADR Tweets dataset has a very high standard deviation of $\pm 2.65\%$. We observe that ASWA and NASWA are almost always able to achieve higher performance with a very low standard deviation. This suggests that both ASWA and NASWA are extremely stable when compared to the standard model. They significantly improve the robustness, on an average, by 72% relative to the original model and on Diabetes (MIMIC), a binary classification dataset, by 89% (relative improvement). We observe similar results for the LSTM based models in Table 3.

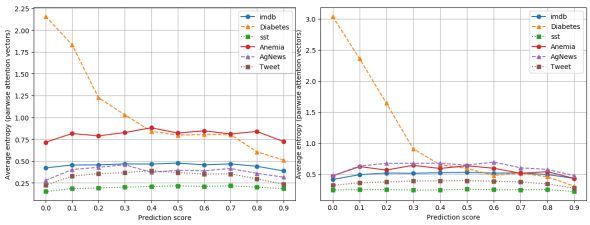


(a) CNN models (b) LSTM models

Figure 3: Prediction's standard deviation for CNN and LSTM based models for all binary classification datasets under consideration. Predictions are bucketed in intervals of size 0.1, starting from 0 (containing predictions from 0 to 0.1), until 0.9

We further analyze the prediction score stability by computing the mean standard deviation over the binned confidence intervals of the models in Figure 3a. We note that on an average, the standard deviations are on the lower side. However, we observe that the mean standard deviation of the bins close to 0.5 is on the higher side as is expected given the high uncertainty. On the other hand both, ASWA and NASWA based models are relatively more stable than the standard CNN based model. We observe similar behaviours for the LSTM based models in Figure 3b. This suggests that our proposed methods, ASWA and NASWA, are able to obtain relatively better stability without any loss in performance. We also note that both ASWA and NASWA had relatively similar performance over more than 10 random seeds.

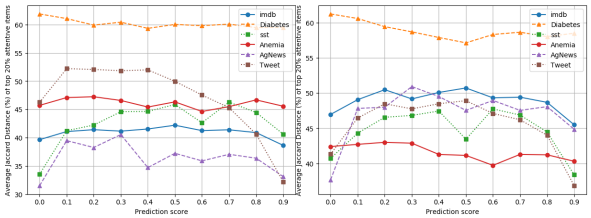
5.2 Attention Stability



(a) CNN models (b) LSTM models

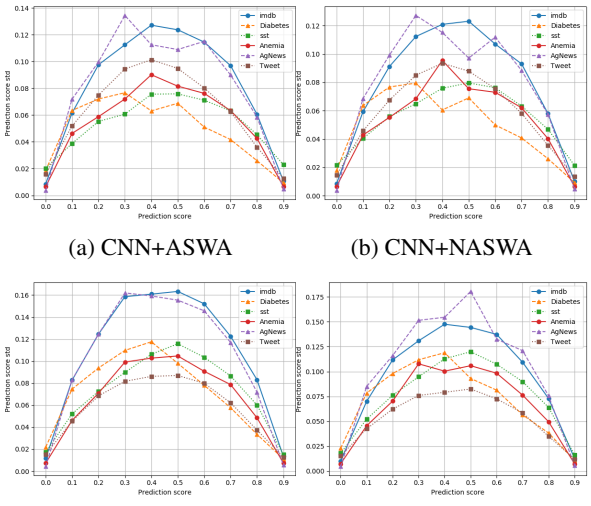
Figure 4: Average attention entropy against the bucketed predictions for CNN and LSTM based models. Figure highlights the high entropy between attention based distributions from differently seeded models (especially for the Diabetes-MIMIC datatset), indicating towards model instability.

We now consider the stability of attention distributions as a function of random seeds. We first plot the results of the experiments for *standard* CNN based binary classification models over uniformly binned prediction scores for positive labels in Figure 4a. We observe that, depending on the datasets, the attention distributions can become extremely unstable (high entropy). We specifically highlight the Diabetes(MIMIC) dataset's entropy distribution. We observe similar, but relatively worse results for the LSTM based models in Figure 4b. In general, we would expect the entropy distribution to be close to zero however, this doesn't seem to be the case. This means that using attention distributions to interpret models may not be reliable and can lead to misinterpretations.



(a) CNN models (b) LSTM models

Figure 5: Jaccard distance highlighting instability in attention distributions of CNN and LSTM based models.

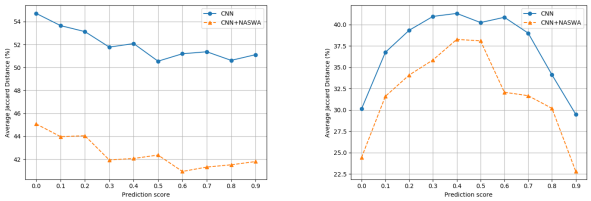


(a) CNN+ASWA (b) CNN+NASWA

(c) LSTM+ASWA (d) LSTM+NASWA

Figure 6: Improved prediction stability from ASWA and NASWA for CNN and LSTM based models

We use the top 20% of the most important items (indices) in the attention distribution for each dataset over 10 runs and plot the Jaccard distances for CNN and LSTM based models in Figure 5a and Figure 5b. We again notice a similar trend of unstable attention distributions over both CNN and LSTM based attention distribution.



(a) Diabetes (b) SST

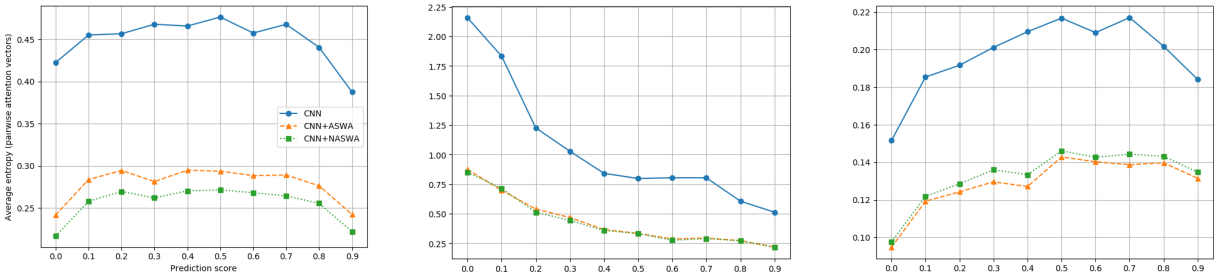
Figure 7: Gradient based interpretations' stability improvement from NASWA on CNN based models. The Jaccard distance is calculated using the top 20% attentive items.

In the following sections for space constraints, we focus on CNN based models with additive attention. Our results on LSTM based models are provided in the attached supplementary material. We note that the observations for LSTM models are, in most cases, similar to the behaviour of the CNN based models. Scaled dot-product based models are also provided in the supplementary material and we notice a similar trend as the additive attention.

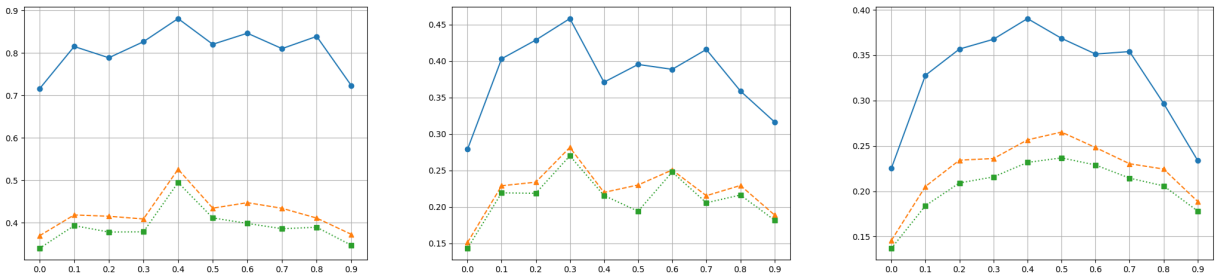
We now focus on the effect of ASWA and NASWA on binary and multi-class CNN based neural models separately.

Binary Classification In Figure 8, we plot the results of the models with ASWA and NASWA. We observe that both these algorithms significantly improve the model stability and decrease the entropy between attention distributions. For example, in Figure 8b, both ASWA and NASWA decrease the average entropy by about 60%. We further notice that NASWA is slightly better performing in most of the runs. This empirically validates the hypothesis that averaging the weights from divergent weights (when the norm difference is higher than the average norm difference) helps in stabilizing the model's parameters, resulting in a more robust model.

Multi-class Classification In Figure 9, we plot the entropy between the attentions distributions of the models for the SNLI dataset (CNN based model), separately for *each label* (*neutral*, *contradiction*, and *entailment*). We notice, similar observations as the binary classification models, the ASWA and NASWA algorithms are able to significantly improve the entropy of the attention distributions and increases the robustness of the model

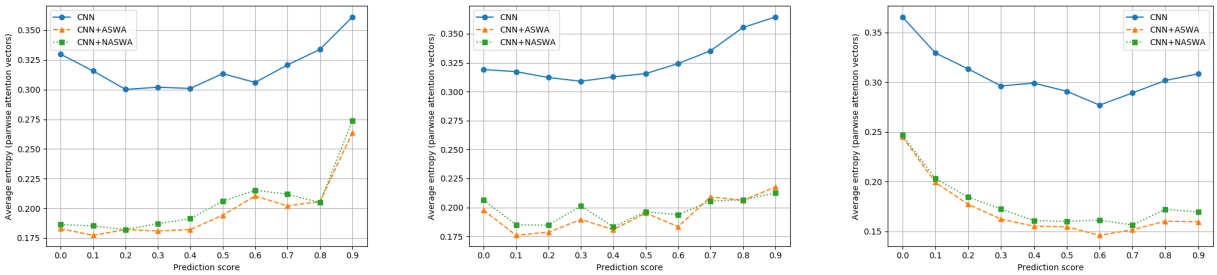


(a) IMDB (b) Diabetes (c) SST



(d) Anemia (e) AgNews (f) ADR Tweets

Figure 8: Attention stability improvement from ASWA and NASWA on CNN based models.



(a) Label 0 prediction vs entropy (b) Label 1 prediction vs entropy (c) Label 2 prediction vs entropy

Figure 9: Attention stability improvement from ASWA and NASWA on CNN based model for the SNLI dataset.

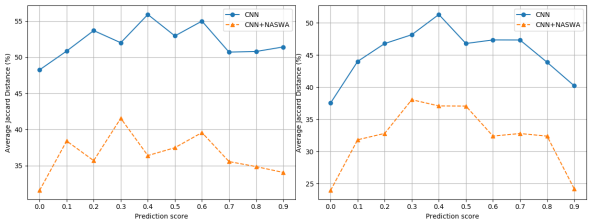
with random seeds.

5.3 Gradient-based Interpretations

We now look at an alternative method of interpreting deep neural models and look into the consistency of the gradient-based interpretations to further analyze the model's instability. For this setup, we focus on binary classifier and plot the results on the SST and the Diabetes dataset in particular since they cover the low and the high end of the entropy spectrum (respectively). We notice similar trends of instability in the gradient-based interpretations from model inputs as we did for the attention distributions. Figure 7 shows that the entropy between the gradient-based interpretations from differently seeded models closely follows the same trend as the attention distributions. This result further strengthens our claim on the importance of model stability and shows that over different runs of the same model with different seeds, we may get different interpretations using gradient-based feature importance. Moreover, Figure 7 shows the impact of ASWA towards making the gradient-based interpretations more consistent, thus, significantly increasing the stability.

5.4 LIME based Interpretations

We further evaluated the surrogate model based interpretability using LIME (Ribeiro et al., 2016). LIME obtains a locally linear approximation of the model's behaviour for a given sample by perturbing it and learning a sparse linear model around it. We focus on AgNews and SST based datasets and obtain interpretability estimates using LIME. Once again, we notice a similar pattern of instability as the other two interpretability methods. In Figure 10 we present our results from the LIME based interpretations with Jaccard distance as the measure. Note that we measure the Jaccard distance over the top 20% most influential items. We



(a) AgNews (b) SST

Figure 10: LIME based interpretations' stability improvement from NASWA on CNN based models. The Jaccard distance is calculated using the top 20% attentive items.

observe once again that NASWA helps in reducing the instability and results in more consistent interpretations.

In all our experiments, we find that a significant proportion of interpretations are dependent on the instantiation of the model. We also note that we perform experiments over 100 random seeds for greater statistical power and see similar patterns[3].

6 Discussion

Recent advances in adversarial machine learning (Neelakantan et al., 2015; Zahavy et al., 2016) have investigated robustness to random initialization based perturbations, however, to our knowledge, no previous study investigates the effect of random-seeds and its connection on model interpretation. Our study analyzed the inherent lack of robustness in deep neural models for NLP. Recent studies cast doubt on the consistency and correlations of several types of interpretations (Doshi-Velez and Kim, 2017; Jain and Wallace, 2019; Feng et al., 2018). We hypothesise that some of these issues are due to the inherent instability of the deep neural models to random-seed base perturbations. Our analysis (in Section 4) leads to the hypothesis that models with different instantiations may use completely different optimization paths. The issue of variance in all black-box interpretation methods over different seeds will continue to persist until the models are fully robust to random-seed based perturbations. Our work however, doesn't provide insights into instabilities of different layers of the models. We hypothesise that it might further uncover the reasons for the relatively lower correlation between different black-box interpretation methods as these are effectively based off on different layers and granularity.

[3]These results are provided in the appendix.

There has been some work on using noisy gradients (Neelakantan et al., 2015) and learning from adversarial and counter-factual examples (Feng et al., 2018) to increase the robustness of deep learning models. Feng et al. (2018) show that neural models may use redundant features for prediction and also show that most of the black-box interpretation methods may not be able to capture these second-order effects. Our proposals show that aggressively averaging weights leads to better optimization and the resultant models are more robust to random-seed based perturbation. However, our research is limited to increasing consistency in neural models. Our approach further uses first order based signals to boost stability. We posit that second-order based signals can further enhance consistency and increase the robustness.

7 Conclusions

In this paper, we study the inherent instability of deep neural models in NLP as a function of random seed. We analyze model performance and robustness of the model in the form of attention based interpretations, gradient-based feature importance and LIME based interpretations across multiple runs of the models with different random seeds. Our analysis strongly highlights the problems with stability of models and its effects on black-box interpretation methods leading to different interpretations for different random seeds. We also propose a solution that makes use of weight averaging based optimization technique and further extend it with norm-filtering. We show that our proposed methods largely stabilize the model to random-seed based perturbations and, on average, significantly reduce the standard deviations of the model performance by 72%. We further show that our methods significantly reduce the entropy in the attention distribution, the gradient-based feature importance measures and LIME based interpretations across runs.

Acknowledgments

We thank Panos Parpas and Emtiyaz Khan for their feedback on an earlier draft of this paper. We thank the anonymous reviewers for their thorough reviews and constructive comments. Pranava Madhyastha kindly acknowledges the support of Amazon AWS Cloud Credits for Research Award, hardware grant from NVIDIA, Anne O'Neill and the Imperial Corporate Partnership Programme.

References

David Alvarez-Melis and Tommi S Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. *arXiv preprint arXiv:1707.01943*.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. 2010. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Shi Feng, Eric Wallace, Alvin Grissom II, Pedro Rodriguez, Mohit Iyyer, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretation difficult. In *Empirical Methods in Natural Language Processing*.

Reza Ghaeini, Xiaoli Z Fern, and Prasad Tadepalli. 2018. Interpreting recurrent and attention-based neural models: a case study on natural language inference. *arXiv preprint arXiv:1808.03894*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Sarthak Jain, Ramin Mohammadi, and Byron C Wallace. 2019. An analysis of attention over clinical notes for predictive tasks. *arXiv preprint arXiv:1904.03244*.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. *CoRR*, abs/1902.10186.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Stephan Mandt, Matthew D Hoffman, and David M Blei. 2017. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907.

Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2015. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*.

Azadeh Nikfarjam, Abeed Sarker, Karen O'connor, Rachel Ginn, and Graciela Gonzalez. 2015. Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, 22(3):671–681.

Boris T Polyak and Anatoli B Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*.

David Ruppert. 1988. Stochastic approximation. Technical report, Cornell University Operations Research and Industrial Engineering.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models

for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Tom Zahavy, Bingyi Kang, Alex Sivak, Jiashi Feng, Huan Xu, and Shie Mannor. 2016. Ensemble robustness and generalization of stochastic deep learning algorithms. *arXiv preprint arXiv:1602.02389*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Studying Generalisability Across Abusive Language Detection Datasets

Steve Durairaj Swamy and **Anupam Jamatia**
Department of Computer Science
National Institute of Technology
Agartala, India
{steve050798,anupamjamatia}@gmail.com

Björn Gambäck*
Department of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway
gamback@ntnu.no

Abstract

Work on Abusive Language Detection has tackled a wide range of subtasks and domains. As a result of this, there exists a great deal of redundancy and non-generalisability between datasets. Through experiments on cross-dataset training and testing, the paper reveals that the preconceived notion of including more non-abusive samples in a dataset (to emulate reality) may have a detrimental effect on the generalisability of a model trained on that data. Hence a hierarchical annotation model is utilised here to reveal redundancies in existing datasets and to help reduce redundancy in future efforts.

1 Introduction

With the growth of the internet and the increasingly smaller barrier to entry, social media have become viable platforms for people to make their views known. These easily accessible fora for discourse have given a voice to many minorities and individuals to share their stories. The caveat, however, is that these platforms can be misused to spread hate and harass other individuals, which has given birth to terms such as cyberbullying and trolling. Online harassment has been a point of criticism levied against social media giants such as Facebook and Twitter, who have come under increased pressure to address this misuse. To this end, they have ensured that their community guidelines explicitly ban the usage of profanity/hate speech to harass and bully individuals.

The detection of Online Abuse has proven to be a layered and complex issue. For example, profanity is often treated as a sign of hate speech or offensive language, but profanity can also be used in a wide variety of expressive ways to convey informality, humour, and emphasis. This usage of profanity outside of abuse/insults, coupled with implicit insults that may not contain any profanity, makes the task of classifying abuse online a balancing act of sorts, forming the crux of what makes this task hard to tackle: stricter guidelines may hamper a well-meaning individual's freedom of speech, while more lenient guidelines may empower those who exploit them.

As it stands, the intricacies of free speech do not translate well to machine understanding. This has led to the continued use of human moderators in the abusive language detection space. Content is flagged by users, reviewed by a human and removed if it violates the platform's community guidelines. The main problem with this system is the sheer volume of content to be reviewed, giving human moderators very little time to arrive at a decision. Another issue that was highlighted by Roberts (2019) is the impact that reviewing online abuse can have on a worker's mental well-being. These issues have led to many social media giants, such as Facebook, to seek machine learning-based solutions — to replace or supplement the current human moderator system.

Automatic detection of abusive language online can be seen as a union of the plethora of subtasks that have been tackled: Cyberbullying, Hate Speech (also further constrained as racism, sexism, and harassment of particular minorities), Trolling, etc. Research in the field tends to focus on one of the particular subtasks. It has been argued by some (Schmidt and Wiegland, 2017; Waseem et al., 2017b) that due to this phenomenon where works tackle restricted subsets of abusive language, it has become difficult to make judgements about whether the features being used can perform well in other subtasks of abusive language detection — as they are often only evaluated on a single dataset, specific to one domain and subtask, and annotated in a specific way.

*Also at: RISE SICS, Kista, Sweden.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 940–950
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

Waseem et al. (2017b) proposed that there exists an overlap between these subtasks and subsequently proposed a typology that emphasises identifying the target of abuse and whether the abuse is implicit or explicit. Their typology could potentially be applied to all stages of system development, from data collection to the final model building. This, they hoped, would help to synthesise the different subtasks. This idea was expanded upon in the Offensive Language Identification Dataset (OLID; Zampieri et al. 2019a) to a hierarchical, three-level annotation model.

After further discussing related research in the next section, this work looks at various publicly available datasets in the field (Section 3), and performs both in-domain (Section 4) and cross-dataset training and testing to observe whether models trained on one dataset generalise well when tested against other datasets (Section 5). It also makes some qualitative assessments on why models trained on specific datasets generalise better than others. Additionally, the OLID dataset based on the typology by Waseem et al. (2017b) is used to observe whether the hierarchical annotation model is sufficient to synthesise the various subtasks of abusive language detection. To this end, experiments were run using BERT, Bidirectional Encoder Representations from Transformers (Devlin et al., 2018), to compare its performance to other popular models that have been used for abusive language detection (Section 6).

2 Previous Work

Abusive language detection has served as an umbrella term for a wide variety of subtasks. Research in the field has typically focused on a particular subtask: Hate Speech (Davidson et al., 2017; Founta et al., 2018; Gao and Huang, 2017; Golbeck et al., 2017), Sexism/Racism (Waseem and Hovy, 2016), Cyberbullying (Xu et al., 2012; Dadvar et al., 2013), Trolling and Aggression (Kumar et al., 2018a), and so on. Datasets for these tasks have been collected from various social media platforms, such as Twitter (Waseem and Hovy, 2016; Davidson et al., 2017; Founta et al., 2018; Burnap and Williams, 2015; Golbeck et al., 2017), Facebook (Kumar et al., 2018a), Instagram (Hosseinmardi et al., 2015; Zhong et al., 2016), Yahoo! (Nobata et al., 2016; Djuric et al., 2015; Warner and Hirschberg, 2012), YouTube (Dinakar et al., 2011), and Wikipedia (Wulczyn et al., 2017), with annotation typically carried out on crowdsourcing platforms such as CrowdFlower (Figure Eight)[1] and Amazon Mechanical Turk.[2]

All these datasets represent multi-class classification problems, with the exception of the Kaggle's Toxic Comment Classification challenge,[3] which entails multi-label classification, and OLID (Zampieri et al., 2019a) used in the SemEval-2019 'OffensEval' shared task (Zampieri et al., 2019b), which builds on a hierarchical annotation model (Hierarchy of Multi-Class Classifiers).

Choice of features has been the crucial difference between the various approaches to abusive language detection. For the most part, word-level n-grams have been highly predictive, with other linguistic features such as part-of-speech tags (Xu et al., 2012; Davidson et al., 2017) and sentiment score (Van Hee et al., 2015; Davidson et al., 2017) providing slight improvements. Due to their ability to perform better in an online setting where spelling errors and adversarial behaviour are commonplace, character-level features have been endorsed (Mehdad and Tetreault, 2016), and also shown to often be superior to word-level information for this task (Meyer and Gambäck, 2019). Metadata about users have also been used as features: Waseem and Hovy (2016) claim gender information leads to improved performance, while Unsvåg and Gambäck (2018) report user-network data to be more important. Schmidt and Wiegland (2017) provides a comprehensive overview of many of the features used and their efficacy.

In terms of models, popular classical classification approaches include Logistic Regression and LSVM (Linear Support Vector Machines). Deep Neural networks such as Convolutional Neural Networks, CNN (Zhang et al., 2018; Gambäck and Sikdar, 2017) and variations of Recurrent Neural Networks, RNN (Pitsilis et al., 2018; Gao and Huang, 2017) have seen widespread success, regularly obtaining state-of-the-art results on various datasets. Lee et al. (2018) used the Founta et al. (2018) dataset to conduct a comparative study of the performance of many popular models. In the 'OffensEval' shared task (Zampieri et al., 2019b), the use of contextual embeddings such as BERT (Devlin et al., 2018) and ELMo (Peters et al., 2018) exhibited the best results.

[1] figure-eight.com
[2] mturk.com
[3] bit.ly/2HNfLaB

Generalisability of a model has also come under considerable scrutiny. Works such as Karan and Šnajder (2018) and Gröndahl et al. (2018) have shown that models trained on one dataset tend to perform well only when tested on the same dataset. Additionally, Gröndahl et al. (2018) showed how adversarial methods such as typos and word changes could bypass existing state-of-the-art abusive language detection systems. They also observed unimpressive results when using ULMFiT (Howard and Ruder, 2018) for abusive language detection, but argued that model architecture is less important than the type of data and the annotation scheme.

Karan and Šnajder (2018) experimented with cross-domain training and testing, and opted to use the same model (LSVM) with minimal features and to preprocess in favour of interpretability. They also reported positive improvements using Frustratingly Easy Domain Adaptation (FEDA; Daumé III, 2007) to augment smaller datasets with larger ones. Fortuna et al. (2018) concurred, stating that although models perform better on the data they are trained on, slightly improved performance can be obtained when adding more training data from other social media. Similarly, Waseem et al. (2018) attempted to address the problem of differences between datasets by building a robust multi-task learning model, which improves upon single-task performance by using auxiliary samples from select datasets. Their work revealed that such models could be competitive with the state-of-the-art single-task models with the additional benefit of allowing prediction on other datasets as well. This helps in negating hidden biases within datasets and promoting generalisability.

3 Datasets

The experiments in the next section will be based on four different datasets, annotated for hate speech and/or offensive language, as described below. The social media platform of choice, Twitter, was selected due to the availability of a multitude of easy to access datasets. The datasets are all in English and from Twitter, and largely chosen based on popularity and availability.

The first two datasets, from Waseem and Hovy (2016) and from Davidson et al. (2017) were chosen due to their widespread use as benchmarks for models. The third, from Founta et al. (2018) was selected because of its large size, while the fourth (Zampieri et al., 2019a) was included since it is using the contemporary hierarchical model. Some other large datasets were discarded since they are either not from Twitter (such as the Kaggle Toxicity classification of Wikipedia comments, Wulczyn et al., 2017) or not easily or openly available (e.g., Silva et al., 2016; Golbeck et al., 2017).

3.1 The Waseem and Hovy Dataset

In their work on the disambiguation of types of hate speech, Waseem and Hovy (2016) released a dataset of 16,914 tweets. They solicited their tweets using a lexicon of hate speech terms, and manually annotated them with three tags: `racism`, `sexism`, and `none`. Waseem and Hovy used an expert outside annotator for reviewing their annotations to mitigate any bias. The database is provided as a set of tweet IDs with tags, but many of the actual tweets have been removed over time, in particular those belonging to the racist class.[4] The first set of rows in Table 1 describes the dataset, including a comparison of the original Waseem and Hovy (2016) dataset to the one available for download using the Twitter API when the present experiments were initiated.

3.2 The Davidson et al. Dataset

Davidson et al. (2017) made publicly available a Twitter dataset with three labels: `hate_speech`, `offensive_language`, and `neither`. Similar to Waseem and Hovy (2016), they used a lexicon of hate speech terms derived from `Hatebase.org` and queried Twitter using these terms to collect potentially hateful tweets. Each tweet was annotated by at least three CrowdFlower workers and the tags were assigned based on the majority decisions. The final dataset available online contains 24,783 tweets. Table 1 provides some statistics of the dataset, which henceforth will be referred to as the Davidson et al. dataset.

Note the very large fraction of abusive tweets in the dataset. A possible explanation for this was given by Waseem et al. (2018), who noted that 2,161 tweets in Davidson et al.'s dataset written in African American Vernacular English had been annotated as offensive or hateful when including the n-word, although the actual usage was to mark group inclusion and informality. While Waseem et al. discuss that these errors were due to the

[4]Note that this discrepancy means that comparisons to work by others on this dataset are not straight-forward.

Dataset	Total	Normal	Hatespeech			Offensive / Abusive					Spam
				racism	sexism						
Waseem and Hovy											
original	16,914	11,559	5,355	1,972	3,383	N/A					N/A
available	11,112	8,185	2,927	17	2,910	N/A					N/A
Davidson et al.	24,783	4,163	1,430			19,190					N/A
Founta et al.	99,996	53,851	4,965			27,150					14,030
							UNT	TIN (targeted)			
								IND	GRP	OTH	
Zampieri et al.	14,100	9,460	N/A			4,640	551	2,507	1,152	430	N/A

Table 1: Overview of the datasets by Davidson et al., Founta et al., Waseem and Hovy, and Zampieri et al.

scarcity of African Americans among the annotators, they could also be attributed to lack of meta-information about the tweet authors: had the annotators known that those tweets were written by African Americans, they would probably have induced that the n-word was not used offensively.

3.3 The Founta et al. Dataset

Founta et al. (2018) released a large Twitter dataset with four labels: `hateful`, `abusive`, `normal`, and `spam`. The main part of their work revolved around a methodology to collect and annotate data over crowdsourcing platforms. They collected tweets from the Live Twitter stream and filtered them using sentiment score (searching for tweets with strong negative polarity) and a lexicon of offensive words from `Hatebase.org` and `noswearing.com/dictionary`.

Table 1 also introduces the Founta et al. dataset, which with a total of $99,996$ tweets is by far the largest in the present study, but also contains a sizable fraction of spam tweets (a category which is not included in the other datasets).

3.4 OLID

The Offensive Language Identification Dataset, OLID (Zampieri et al., 2019a) was used in SemEval-2019 Task 6: 'OffensEval' (Zampieri et al., 2019b). It consists of $14,100$ tweets annotated through a unique hierarchical model whose basic idea was proposed by Waseem et al. (2017b). For the shared task, the data was split into (non-stratified) training and test sets containing $13,240$ and 860 tweets, respectively.

As can be seen in last rows of Table 1, there are three annotation levels in OLID, each of which was directly reflected as a subtask in OffensEval:

A. Whether the tweet can be classified as being offensive (`OFF`) or non-offensive (`NOT`).

B. Tweets labelled as `OFF` are further classified as either `UNT` (untargeted insult/abuse) or `TIN` (targeted insult/abuse).

C. Tweets labelled as `TIN` are sub-divided as `IND` (insults targeted at an individual), `GRP` (insults targeted at a minority group) or `OTH` (insults targeted at an issue or organisation).

4 Preliminary Feature and Model Study

The first set of experiments aimed to test the efficacy of BERT (Devlin et al., 2018) when tackling the Abusive Language Detection task. For this, BERT's performance was compared to three other popular classifiers: Linear SVM, an LSTM (Long Short-Term Memory) Recurrent Neural Network (Hochreiter and Schmidhuber, 1997), and ELMo (Peters et al., 2018). The methodology and models are briefly explained here.

To shed some light on the models themselves rather than the features, no extra surface-level features or linguistic features were utilised in the classification. Also preprocessing was minimal, with lower-casing of tweets being the only standard. However, fine-tuning was carried out on the models' hyper-parameters, such as sequence length, drop out, and class weights. Test and training sets were created for each dataset by performing a stratified split of 20% vs 80%, with the larger part used for training the models. The training sets were further subdivided, keeping 1/8 shares of them as separate validation sets during development and fine-tuning of the hyper-parameters. However, the validation sets were conflated with the training sets for the final results as some of the datasets were already quite small and the models benefited from the extra data. Information on the models themselves are provided below.

Dataset	LSVM		LSTM		ELMo		BERT	
	Acc.	F_1	Acc.	F_1	Acc.	F_1	Acc.	F_1
Waseem and Hovy	.8911	.5696	.8498	.5312	.8614	.5394	**.9023**	**.5837**
Davidson et al.	.9014	.7278	.9143	.7419	.8909	.6802	**.9172**	**.7727**
Founta et al.	.8034	.6591	.8161	.6788	.8094	.6732	**.8187**	**.6960**
OLID (Subtask A)	.7610	.7068	.7894	.7479	.7663	.7198	**.8004**	**.7738**

Table 2: Model test tesults (macro-F_1 and accuracy) for all datasets; the best performer is in bold.

4.1 Linear SVM

The Linear SVM (LSVM) was modelled and trained in the `Scikit-learn`[5] library (Pedregosa et al., 2011), utilising a TF-IDF vector representation for the tweets. The classes were artificially balanced and overfitting penalised using L2 regularisation. Interesting hyperparameters included the n-gram range and whether to use character or token n-grams. For example, the Davidson et al. dataset tended to perform better with token n-grams, while the Waseem and Hovy dataset worked better with character n-grams. The inclusion of unigrams was also pivotal to good classifier performance when using token n-grams.

4.2 LSTM Network

The tested Deep Learning Model was built on a fairly simple LSTM architecture using Keras[6] with a TensorFlow[7] back end. The 'Adam' optimiser (Kingma and Ba, 2014) was paired with categorical cross-entropy loss function for model training. Again no statistical or linguistic features were used and the only preprocessing involved lower-casing the tweets. The first layer used a 200 dimensional `GloVe embedding`,[8] pre-trained on 2 billion tweets (Pennington et al., 2014), with embedding weights fixed throughout the training. The Embedding Layer was followed by an LSTM layer of 200 units. The final layer was a dense layer with softmax activation and layer size dependent on the number of classes in the dataset being tested. The most significant hyperparameters were found to be dropout and class weights.

4.3 ELMo

The third model tested used ELMo for feature extraction and was implemented in the TensorFlow hub module[9] with 1024 dimensional ELMo embeddings. This input was passed through an LSTM layer of dimension 256 and then a dense layer with a softmax activation function. The size of the last dense layer was again equal to the number of labels that should be classified. The 'Adam' optimiser and categorical cross-entropy loss function were used during training. ELMo's standalone performance was found to not be as impressive as hoped, with the batch size and usage of dropout significantly affecting classification rates.

4.4 BERT

$\texttt{BERT}_{\texttt{base, uncased}}$ was used as the underlying pre-trained model, in a fine-tuning only approach with no statistical or linguistic features. The model built on the `run_classifier` API provided on the BERT GitHub page[10] and the BERT tokeniser, which simply lower-cases sentences and removes illegal characters. $\texttt{BERT}_{\texttt{base,uncased}}$ trains a total of 110 million parameters, and contains 12 transformer blocks and 12 self-attention heads with hidden layer dimension 768. The most successful parameter settings utilised larger maximum sequence lengths, but smaller batch sizes and lower learning rates. The best models used a learning rate of e^{-5} and batch size 32 with varying maximum sequence lengths between 60 and 70. Other parameters worth mentioning are the number of epochs and the Linear Warm-up Proportion.

4.5 Results

The experimental results are recorded in Table 2, with most improvements and decrements in performance across models being minimal. BERT exhibits the best results for all datasets used in the experiments (with a significance level of 0.05). Surprisingly, ELMo was neither competitive with BERT nor with the GLoVE-embedding LSTM recurrent neural network (when tested with the same statistical significance level).

[5] `scikit-learn.org/stable/`
[6] `github.com/fchollet/keras`
[7] `tensorflow.org/`
[8] `nlp.stanford.edu/projects/glove/`
[9] `tfhub.dev/google/elmo/2`
[10] `github.com/google-research/bert`

Dataset	Positive labels	Negative labels	Positive label fraction
Waseem and Hovy	racism, sexism	neither	26.34%
Davidson et al.	hate_speech, offensive_language	neither	77.43%
Founta et al.	hateful, abusive	spam, none	32.12%
OLID	OFF	NOT	32.91%

Table 3: Cross-dataset experiment, positive and negative label split.

Dataset	Waseem and Hovy		Davidson et al.		Founta et al.		OLID	
	Acc.	F_1	Acc.	F_1	Acc.	F_1	Acc.	F_1
Waseem and Hovy	**.9037**	**.8755**	.5626	.5296	.7205	.5824	.6716	.5982
Davidson et al.	.7719	.6928	**.9639**	**.9351**	.9261	.9157	.7514	.6847
Founta et al.	.7278	.6049	.8324	.7559	**.9421**	**.9340**	.7862	.7447
OLID	.7108	.6269	.8247	.7308	.9251	.9162	**.8004**	**.7738**

Table 4: Cross-dataset test results (accuracy and macro-F_1) for all dataset combinations, using the BERT models. Rows show the dataset used to train the model and columns the dataset used for testing.

5 Cross-Dataset Training and Testing

In the second round of experiments, the best models built for individual dataset were used to test generalisability across the other datasets. For all datasets, these were BERT models, but with varying hyper-parameter settings. Karan and Šnajder (2018) used a simpler Linear SVM model for all the datasets for the sake of interpretability, while the aim here, in contrast, was to see how well the best models (that may have learnt some dataset-specific biases) performed on other datasets. This was done to investigate how well state-of-the-art systems perform in a real-life scenario, i.e., when exposed to data from other domains, with the hypothesis that a model trained on one dataset that exhibits comparatively reasonable results on other datasets can be expected to generalise well.

For these experiments, the models were tested on the test set which had been generated for the preliminary model study described in Section 4. As there exists a large number of heterogeneous annotation schemes between datasets, the same approach as Karan and Šnajder (2018) was taken, separating the tags in each dataset according to positive (abusive) and negative (benign) labels. This separation is represented in Table 3, which also gives the percentage of positive samples in each dataset. As can be seen, three of the datasets contain slightly less than 1/3 abusive instances. The Davidson et al. dataset stands out, by containing 3/4 offensive instances. As discussed in Section 3.2, this can probably be attributed to how those tweets were selected and annotated.

The results of cross-dataset testing are presented in Table 4. Considerable performance drops can be observed when going from a large training dataset to a small test set (i.e., Founta et al.'s results when tested on the Waseem and Hovy dataset) and vice versa. This is in line with a similar conclusion by Karan and Šnajder (2018).

It is surprising to see how well a model trained on Founta et al.'s dataset performs when tested on OLID (Zampieri et al., 2019a) and vice versa. However, this can be expected to be the case where there is a good agreement between the datasets, i.e., there is a large amount of similar data shared between them. To this effect, the Founta et al. dataset was searched with terms used by Zampieri et al. when collecting data for OLID, giving around $6,600$ hits. For comparison, OLID gets around $12,200$ hits with the same set of terms.

The most interesting observation is that datasets with larger percentages of positive samples tend to generalise better than datasets with fewer positive samples, in particular when tested against dissimilar datasets. For example, we see that the models trained on the Davidson et al. dataset, which contains a majority of offensive tags, perform well when tested on the Founta et al. dataset, which contains a majority of non-offensive tags. (The differences are all statistically significant when the test set is Waseem and Hovy.) Similar trends were observed by Karan and Šnajder (2018) when employing the Kolhatkar et al. (2018) and TRAC-1 (Kumar et al., 2018a) datasets, that have 62.7% and 56.6% positive samples, respectively, and exhibited better results in cross-dataset testing than datasets with lower positive sample ratios.

	Subtask A		Subtask B		Subtask C	
	BERT	Top	BERT	Top	BERT	Top
F_1	.8168	.8286	.6997	.7545	.6162	.6597
Acc.	.8546	.8628	.9000	.9250	.7136	.7277
Rank	2	1	9	1	6	1

Table 5: BERT test set results (macro-F_1 and accuracy) compared to top OffensEval shared task performers.

6 Synthesising Subtasks Using the Hierarchical Model

The OLID dataset was used to perform cross-dataset training and testing similar to the experiments of the previous section. However, since OLID uses a hierarchical annotation model (differing from the annotation schemes of the other datasets), this task was approached from a different angle. A model trained on the three subtasks of the OLID dataset (described at the end of Section 3.4) was tested for the task of tagging the in-domain Twitter datasets. This makes it possible to not only see how well OLID-trained models generalise to other data, but to identify the overlap between the different subtasks that the other datasets tackle by observing what percentage of documents under each subtask share common OLID tags.

For the OLID classifiers, a BERT model was used without any extra statistical features and with minimal preprocessing (only lower-casing of tweets). The classifiers were then fine-tuned to the different subtasks, again showing a positive correlation between sequence length and classifier performance. For the results to be comparable to those obtained in the OffensEval 2019 shared task, the same test set was used as in that task. Model performances are reported in Table 5, along with what rank the model would have obtained if it had been submitted to OffensEval 2019, showing that the models are competitive when compared to the top shared task submissions.

The tested model was trained for a total of 3 epochs with a batch size of 16 and learning rate e^{-5}. The maximum sequence length was set to 70 for subtasks A and C, but to 60 for subtask B, where over-fitting was observed on sequence length 70. Also in subtask C the model showed significant signs of over-fitting, with the BERT approach only achieving an F_1 score of 0.52. In this case, a technique was borrowed from the top subtask C submission to OffensEval (Radivchev and Nikolov, 2019), namely to use lower decision boundaries for the `OTH` (0.2) and `GRP` (0.3) tags, instead of the typical decision boundary probability of 0.5. As can be seen in the table, this addition led to huge improvements ($F_1 = 0.62$), compared to the models using the typical decision boundary ($F_1 = 0.52$), although the achieved scores still were not close to the top submission.

Returning to the tagging/synthesis experiments, the entire datasets were used. The results are presented in Table 6. Here we see quite a bit of overlap between the offensive and hate speech tags with the majority tag being `(OFF, TIN, IND)` by a landslide. Clearly, these results can become trivial if the differences boil down to whether the model generalises well to the other datasets used here. This is why only in-domain (Twitter datasets) are considered here and the results also are discussed while taking this into account.

In the Davidson et al. dataset, the non-abusive tag, `neither` had a much lower percentage of its tweets annotated under `NOT` (69.37%) by the OLID classifier when compared to other datasets. This observation may be attributed to the data collection techniques used by Davidson et al., who filtered tweets based on a hate speech lexicon before annotating them, as well as to profanities occurring within the `neither` tag, causing a dip in the amount of explicitly non-offensive tweets.

A similar issue is seen, but to a lesser extent, in the `neither` tag of the Waseem and Hovy dataset, which also was extended by using a sample of hateful tweets. Another interesting observation with that dataset is that the majority class for the `sexism` tag in Subtask A was `NOT`. This complies with observations by both Waseem and Hovy and Davidson et al. (2017) that the human coders considered sexist terms as offensive rather than hateful. However, in terms of our classifier, this may only be due to the implicit nature of most sexist insults and a lack of sexist samples within the OLID dataset. Founta et al.'s dataset shows a high number of `hateful` tweets classified as `NOT`, which may be due to the implicit nature of sexism or sarcasm in the tweets involved.

Dataset	Tag	Subtask A		Subtask B		Subtask C		
		OFF	NOT	UNT	TIN	IND	GRP	OTH
Waseem and Hovy	racism	**52.94**	47.06	0.00	**100**	**64.70**	23.52	11.76
	sexism	42.96	**57.04**	8.28	**91.72**	**54.06**	30.27	15.67
	neither	20.22	**79.78**	28.27	**71.73**	**67.11**	25.95	6.94
Davidson et al.	hate_speech	**84.13**	15.87	3.35	**96.65**	**63.22**	24.61	12.17
	offensive_language	**86.89**	13.11	7.45	**92.55**	**71.89**	20.39	7.72
	neither	30.63	**69.37**	20.44	**79.56**	**63.29**	5.48	31.23
Founta et al.	hateful	**78.11**	21.89	5.92	**94.08**	**52.35**	27.37	20.28
	abusive	**97.34**	2.66	26.04	**73.96**	**75.56**	10.42	14.02
	normal	9.23	**90.77**	24.82	**75.18**	**60.24**	35.84	3.92
	spam	8.72	**91.28**	43.59	**56.41**	**50.98**	1.71	47.31
Zampieri et al.	*(actual annotation fraction)*	*32.91*	*67.09*	*11.88*	*88.12*	*61.31*	*28.17*	*10.52*

Table 6: Results of using a BERT model trained on OLID to tag other the datasets, for each OffensEval subtask. The values are percentages of tweets in each class (rows) annotated with the corresponding OLID tag (columns). Note that in the version of the Waseem and Hovy dataset used here, the racism tag only had 17 samples.

Some blanket statements that can be made given these results are that hate speech is highly targeted, mainly at individuals, but with a significant share targeted at groups and other institutions/issues. Offensive language, on the other hand, tends to be highly targeted only at individuals. Furthermore, the dearth of data belonging to the UNT, GRP and OTH tags may have had a detrimental effect on the model leading to the lob-sided (OFF, TIN, IND) classification.

7 Discussion and Conclusion

The paper makes two major contributions: First, an evaluation of the general effectiveness of BERT in Abusive Language Classification tasks and its ability to obtain results comparable to — or better than — the state-of-the-art by only fine-tuning.

Second, experiments showing that datasets with larger percentages of positive samples generalise better than datasets with fewer positive samples when tested against a dissimilar dataset (at least within the same platform, e.g., Twitter), which indicates that a more balanced dataset is healthier for generalisation. This observation should be accounted for when attempting to build new datasets to tackle Abusive Language Detection, but this is far from the only problem faced when attempting to create such datasets.

Looking at the various available datasets in this field, it is obvious that it cannot be expected for a single dataset to encompass all facets of abuse online. For example, on scanning the OLID (Zampieri et al., 2019a) using a lexicon of sexist and racist terms from Hatebase.org only a measly 55 and 567 hits, respectively, were obtained. Armed with this information we cannot possibly expect a model trained on the OLID dataset to effectively detect racism and sexism online. In fact, most of the data in OLID seem to be political, indicating that it in contrast has a high potential to detect such phenomena.

The point made here is that datasets used in the Abusive Language Detection space must be more representative of all facets of abusive language, if we expect them to generalise to any subset of abuse. Also, there are very few datasets that provide a large number of samples that can be taken advantage of by huge neural networks (Lee et al., 2018). However, we do acknowledge the difficulty in collecting abusive samples as most discourse online is benign. To address these issues, all datasets must advertise the subset of the abusive language they represent. In addition, more work must be done to identify similarities and holes in the representation of datasets. Merging of datasets may also prove to be a promising solution to the non-generalisability problem. Waseem et al. (2018)'s multi-task learning model can be a solid starting point for such endeavours.

A more ambitious solution could be the development of pre-trained embeddings (at the word and/or character level) for Abusive Language Detection, although the procurement of enough broad spanning data to produce a high-quality embedding could again be quite a challenging task.

In terms of whether the hierarchical annotation model helps in reducing redundancy and overlap in Abusive Language Detection subtasks, the answer is both yes and no:

- yes, the hierarchical annotation model does reveal the overlap in the subtasks of abusive language detection; but,

- no, it could hardly be a replacement for the existing multi-class annotation schema.

This is because there is still value in identifying whether a sample is racist / sexist / cyberbullying over just recognising whether the abuse is explicit or not, and in identifying the target of abuse.

However, the hierarchical model in its current form still cannot differentiate between various subsets of abusive language. Future hierarchical models could address this either by adding more levels to further differentiate the subsets or by creating additional levels to identify subsets more explicitly. For example, after the first level of the OLID (Zampieri et al., 2019a) annotation schema, it could branch out into a layer that classifies samples as hate speech, bullying / trolling or as non-abusive use of offensive language. The hate speech tag could then be expanded into another level classifying hate speech as being, e.g., racism, sexism, or other. This way of moving from coarse-grained tags to increasingly finer-grained ones might be a workable approach to tackling hierarchical annotation.

Other issues such as the adversarial methods used to bypass detection methods (Gröndahl et al., 2018) also plague this problem space. Character-based features alleviate this complication to some degree, but more work needs to be done to solve this. Research in this domain has also largely constrained itself to text, while real-world scenarios are quite different — there is a huge section of abuse online that rely on other forms of communication such as images, videos and gifs.

An overall conclusion is that the data is more important than the model when tackling Abusive Language Detection. Schmidt and Wiegland (2017) expressed the need for a benchmark dataset for abusive language tasks, but it would be unwise to say any current dataset fills this role. Future work must focus more on how models generalise to the real world by modifying the testing procedure. A model's performance on the dataset it was trained on cannot be indicative of how well it would perform in a real-life application, and a dataset's quality must be measured on how broad spanning and how representative it is of abusive language as a whole.

Acknowledgments

Thanks to all the researchers who have made their datasets available, specially Waseem and Hovy, Davidson et al., Founta et al., and Zampieri et al., the organisers of SemEval-2019 Task 6: OffensEval ('Identifying and Categorizing Offensive Language in Social Media').

Special thanks to the anonymous reviewers whose comments helped to improve the paper.

References

Pete Burnap and Matthew L. Williams. 2015. Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet*, 7(2):223–242.

Maral Dadvar, Dolf Trieschnigg, Roeland Ordelman, and Franciska de Jong. 2013. Improving cyberbullying detection with user context. In *Advances in Information Retrieval: 35th European Conference on IR Research*, pages 693–696. Springer, Moscow, Russia.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic. ACL.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International Conference on Web and Social Media*, pages 512–516, Montréal, Canada. AAAI Press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Karthik Dinakar, Roi Reichart, and Henry Lieberman. 2011. Modeling the detection of textual cyberbullying. In *The Social Mobile Web: Papers from the 2011 ICWSM Workshop*, pages 11–17, Barcelona, Spain. AAAI Press.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, pages 29–30, Florence, Italy. ACM.

Darja Fišer, Ruihong Huang, Vinodkumar Prabhakaran, Rob Voigt, Zeerak Waseem, and Jacqueline Wernimont, editors. 2018. *Proceedings of the 2nd Workshop on Abusive Language Online*. ACL, Brussels, Belgium.

Paula Fortuna, José Ferreira, Luiz Pires, Guilherme Routar, and Sérgio Nunes. 2018. Merging datasets for aggressive text identification. In (Kumar et al., 2018b), pages 128–139.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of Twitter abusive behavior. *CoRR*, abs/1802.00393.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In (Waseem et al., 2017a), pages 85–90.

Lei Gao and Ruihong Huang. 2017. Detecting online hate speech using context aware models. In *Proceedings of the 11th International Conference on Recent Advances in Natural Language Processing*, pages 260–266, Varna, Bulgaria. INCOMA Ltd.

Jennifer Golbeck, Zahra Ashktorab, Rashad O. Banjo, Alexandra Berlinger, Siddharth Bhagwan, Cody Buntain, Paul Cheakalos, Alicia A. Geller, Quint Gergory, Rajesh Kumar Gnanasekaran, Raja Rajan Gunasekaran, Kelly M. Hoffman, Jenny Hottle, Vichita Jienjitlert, Shivika Khare, Ryan Lau, Marianna J. Martindale, Shalmali Naik, Heather L. Nixon, Piyush Ramachandran, Kristine M. Rogers, Lisa Rogers, Meghna Sardana Sarin, Gaurav Shahane, Jayanee Thanki, Priyanka Vengataraman, Zijian Wan, and Derek Michael Wu. 2017. A large labeled corpus for online harassment research. In *Proceedings of the 2017 ACM Web on Science Conference*, pages 229–233, Troy, New York, USA. ACM.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. 2018. All you need is "love": Evading hate speech detection. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, pages 2–12, Toronto, Canada. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Homa Hosseinmardi, Sabrina Arredondo Mattson, Rahat Ibn Rafiq, Richard Han, Qin Lv, and Shivakant Mishra. 2015. Detection of cyberbullying incidents on the Instagram social network. *CoRR*, abs/1503.03909.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Mladen Karan and Jan Šnajder. 2018. Cross-domain detection of abusive language online. In (Fišer et al., 2018), pages 132–137.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Varada Kolhatkar, Hanhan Wu, Luca Cavasso, Emilie Francis, Kavan Shukla, and Maite Taboada. 2018. The SFU opinion and comments corpus: A corpus for the analysis of online news comments. Manuscript, Simon Fraser University, Vancouver, Canada.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018a. Benchmarking aggression identification in social media. In (Kumar et al., 2018b), pages 1–11.

Ritesh Kumar, Atul Kr. Ojha, Marcos Zampieri, and Shervin Malmasi, editors. 2018b. *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying*. ACL, Santa Fe, New Mexico, USA.

Younghun Lee, Seunghyun Yoon, and Kyomin Jung. 2018. Comparative studies of detecting abusive language on Twitter. In (Fišer et al., 2018), pages 101–106.

Jonathan May, Ekaterina Shutova, Aurelie Herbelot, Xiaodan Zhu, Marianna Apidianaki, and Saif M. Mohammad, editors. 2019. *Proceedings of the 13th International Workshop on Semantic Evaluation (SemEval)*. ACL, Minneapolis, Minnesota, USA.

Yashar Mehdad and Joel R. Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303, Los Angeles, California, USA. ACL/SIGDIAL.

Johannes Skjeggestad Meyer and Björn Gambäck. 2019. A platform agnostic dual-strand hate speech detector. In *Proceedings of the 3rd Workshop on Abusive Language Online*, pages 146–156, Florence, Italy. ACL.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153, Montréal, Canada. IW3C2.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar. ACL.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Georgios Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Effective hate-speech detection in Twitter data using recurrent neural networks. *Applied Intelligence*, 48(12):47304742.

Victor Radivchev and Alex Nikolov. 2019. Nikolov-Radivchev at SemEval-2019 Task 6: Offensive tweet classification with BERT and ensembles. In (May et al., 2019), pages 687–691.

Sarah T. Roberts. 2019. *Behind the Screen: Content Moderation in the Shadows of Social Media*. Yale University Press, New Haven, Connecticut, USA.

Anna Schmidt and Michael Wiegland. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the 5th International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. ACL.

Leandro Araújo Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. 2016. Analyzing the targets of hate in online social media. *CoRR*, abs/1603.07709.

Elise Fehn Unsvåg and Björn Gambäck. 2018. The effects of user features on Twitter hate speech detection. In (Fišer et al., 2018), pages 75–86.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Veronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events. In *Proceedings of the 10th Conference on Recent Advances in Natural Language Processing, Proceedings*, pages 672–680, Hissar, Bulgaria. INCOMA Ltd.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the World Wide Web. In *Proceedings of the 2nd Workshop on Language in Social Media*, pages 19–26, Montréal, Canada. ACL.

Zeerak Waseem, Wendy Hui Kyong Chung, Dirk Hovy, and Joel Tetreault, editors. 2017a. *Proceedings of the First Workshop on Abusive Language Online*. ACL, Vancouver, Canada.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017b. Understanding abuse: A typology of abusive language detection subtasks. In (Waseem et al., 2017a), pages 78–84.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In *Proceedings of the North American Chapter of the Association for Computational Linguistics, Student Research Workshop*, pages 88–93, San Diego, California, USA. ACL.

Zeerak Waseem, James Thorne, and Joachim Bingel. 2018. Bridging the gaps: Multi task learning for domain transfer of hate speech detection. In Jennifer Golbeck, editor, *Online Harassment*, pages 29–55. Springer, Cham, Switzerland.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399, Perth, Australia. IW3C2.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 656–666, Montréal, Canada. ACL.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1 (Long and Short Papers), Minneapolis, Minnesota, USA. ACL.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and categorizing offensive language in social media (OffensEval). In (May et al., 2019), pages 75–86.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on Twitter using a convolution-GRU based deep neural network. In *Proceedings of the 15th International Semantic Web Conference*, pages 745–760, Heraklion, Greece. Springer.

Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J. Miller, and Cornelia Caragea. 2016. Content-driven detection of cyberbullying on the Instagram social network. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 3952–3958, New York, New York, USA. AAAI Press.

Reduce & Attribute: Two-Step Authorship Attribution for Large-Scale Problems

Michael Tschuggnall
Universität Innsbruck
michael.tschuggnall@uibk.ac.at

Benjamin Murauer
Universität Innsbruck
b.murauer@posteo.de

Günther Specht
Universität Innsbruck
guenther.specht@uibk.ac.at

Abstract

Authorship attribution is an active research area which has been actively studied for many decades. Nevertheless, the majority of approaches consider problem sizes of a few candidate authors only, making them difficult to apply to recent scenarios incorporating thousands of authors emerging due to the manifold means to digitally share text. In this study, we focus on such large scale problems and propose to effectively reduce the number of candidate authors before applying common attribution techniques. By utilizing document embeddings, we show on a novel, comprehensive dataset collection that the set of candidate authors can be reduced with high accuracy. Moreover, we show that common authorship attribution methods substantially benefit from a preliminary reduction if thousands of authors are involved.

1 Introduction

Correctly determining the author of an anonymous text has been researched for several decades. Undoubtedly the most groundbreaking work in this area has been conducted in 1964 by Mosteller and Wallace, who attempted to automatically assign the authors of the Federalist Papers by utilizing a simple, but yet efficient statistical approach operating on function words (Mosteller and Wallace, 1964). By showing that writers can indeed be distinguished by their writing style, many approaches have been published in the following years, proposing enhancements by incorporating a variety of so-called stylometric features, methods and learning techniques (Stamatatos, 2009). By categorizing the problem of authorship attribution as a special form of text categorization (Sebastiani, 2002), also the respective methods in terms of different machine learning algorithms are effectively in use. With the advent of deep learning, also several approaches have been proposed recently which utilize comprehensive neural networks. Nevertheless, a recent comparison of all submitted approaches to the cross-domain authorship attribution task at PAN[1] indicates that deep learning is currently not able to surpass traditional methods (Kestemont et al., 2018).

Regardless of the features and methods used, the efficacy of an approach can only be measured by using appropriate datasets. Thereby, the majority of existing approaches focus only on a small number of candidate authors (up to 20, most of the times ten or less, e.g., (Stamatatos, 2009; Juola, 2012)). Only a few studies have examined the performance of authorship attribution approaches on larger amounts of possible authors, e.g., 114 (Madigan et al., 2005), 145 (Luyckx and Daelemans, 2008), 808 (Hitschler et al., 2017) or 1,000 (Shrestha et al., 2017) candidates. To the best of our knowledge, only three studies dive into the multiple thousands of authors: 10,000 (Koppel et al., 2006, 2011) and even 100,000 (Narayanan et al., 2012). While both studies agree that authorship attribution at large scale is substantially more difficult, they still show the potential of performing identification with acceptable accuracy. To be precise, both approaches report good performances only in scenarios where either an *"I don't know"* answer is also accepted (Koppel et al., 2011) or the attribution is not precise, i.e., the statement that the correct author is among the top k ones is sufficient (Narayanan et al., 2012).

Motivated by previous findings, which showed that direct authorship attribution is not feasible in a large scale scenario, we contribute to this field by proposing a two-step approach in this study. Specifically, we propose at first to reduce the number of candidate authors while keeping the correct

[1] PAN is an internationally renowned initiave in the field of digital text forensics and stylometry, https://pan.webis.de

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 951–960
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

author in the reduced set with reasonable accuracy. Incorporating the promising results reported by using embedding representations ((Posadas-Durán et al., 2017)), we also find that a vector space based on document embeddings (Le and Mikolov, 2014) in combination with cosine similarity yields the best results for reducing candidate authors in the large scale. As authorship attribution generally heavily depends on the datasets used (e.g., the text type or the number of training and test documents (Luyckx and Daelemans, 2011; Potthast et al., 2016)) and the datasets used in the mentioned studies are not available,[2] we created a collection of 179 individual, novel datasets using a large question-and-answer (Q&A) network on which we extensively test our models. Using these datasets, we finally also show that a preliminary reduction of candidates substantially improves the overall accuracy of finding the correct author in large settings.

At a glance, our contributions are as follows: (1) We evaluate document embeddings in combination with cosine similarity and show that they outperform n-grams (which have proven to be among the most discriminating features, e.g., Stamatatos, 2013; Kestemont et al., 2018) with respect to the task of reducing the number of candidate authors in large scenarios. Thereby we show that neither n-grams used with similarity measures nor support vector machines are able to keep up with document embeddings. (2) We show that eliminating candidate authors using our approach in large settings—prior to performing direct attribution—substantially improves the accuracy of commonly used attribution methods. (3) We created a novel dataset collection based on a large Q&A network, consisting of 179 sub-datasets, each of which features six up to nearly 20,000 authors. To ensure reproducibility and to encourage further research, we make the dataset publicly available to the research community.

The remainder of this paper is organized as follows: At first, Section 2 summarizes related work and subsequently Section 3 presents the dataset. The proposed approach to reduce candidates using document embeddings and its evaluation is presented in Section 4, while Section 5 shows its impact on direct authorship attribution. Finally, Section 6 concludes and discusses future work.

[2]The dataset used by (Koppel et al., 2011) is partly available (see Section 4.2)

2 Related Work

Features and Methods

In the last decades many different features have been proposed for stylometry problems, which can basically be categorized into lexical, syntactic, structural and other specialized features (Stamatatos, 2009; Stein et al., 2011). For the specific task of authorship attribution, lexical metrics are predominant. Thereby, features are utilized on the character- and word-level, including character/word frequencies (Zheng et al., 2006), average word- and sentence lengths (Grieve, 2007), function word frequencies (Argamon et al., 2003; Zhao and Zobel, 2005), bag-of-words (BOW, Agun and Yilmazel, 2017) or especially character/word n-grams (Sapkota et al., 2015; Stamatatos, 2013; Schwartz et al., 2013 and variants thereof (Stamatatos, 2017). Moreover, derived features such as different readability measures (Tweedie and Baayen, 1998) or compression ratios (Marton et al., 2005) have also been investigated.

Syntactic features include the analysis of (n-grams of) Part-of-Speech (POS) tags (Zhao and Zobel, 2007) or the analysis of the parse tree of sentences (Luyckx and Daelemans, 2008; Tschuggnall and Specht, 2014), whereas structural features analyze indicators like the average paragraph length or the use of indentation (Zheng et al., 2006). In addition, various additional metrics have been proposed, e.g., the analysis of spelling and grammatical errors present in a text (Koppel and Schler, 2003).

From a methodical view, a wide range of machine learning techniques is in use, including Bayesian models, logistic regression, support vector machines (SVM) or decision trees. In most cases, the studies apply multiple classifiers and compare their results (e.g., see the surveys of Stamatatos, 2009; Juola and Stamatatos, 2013; Potthast et al., 2016; Kestemont et al., 2018). Recently, deep learning techniques have also been applied to authorship attribution problems. Thereby, various approaches have been proposed which use convolutional neural networks (CNN, Rhodes, 2015; Shrestha et al., 2017). With respect to input features, embeddings on different levels are heavily utilized, e.g., on words (*word2vec*, Mikolov et al., 2013), documents (*doc2vec*, Le and Mikolov, 2014), or n-grams of characters (Shrestha et al., 2017) or POS-tags (Hitschler et al., 2017). In addition, studies have reported that

embeddings are also highly efficient when fed into common machine learning techniques like SVMs or logistic regression (Agun and Yilmazel, 2017; Posadas-Durán et al., 2017).

In general, it has been shown that especially character n-grams and variants thereof are among the most discriminating features, which perform very well with common machine learning techniques such as out-of-the-box SVMs (e.g., Stamatatos, 2013; Kestemont et al., 2018), ensembles (e.g., Custódio and Paraboni, 2018) as well as with recent deep learning methods (e.g., Shrestha et al., 2017; Rhodes, 2015). Due to this success of n-grams, we chose to rely on them as a reference as is shown in Sections 4 and 5.

Large-Scale Authorship Attribution

As mentioned earlier, the majority of authorship attribution approaches target a relatively small number of candidate authors (up to at most 20). The few studies considering more than a hundred authors utilize various lexical features such as character n-grams together with syntactic features, and achieve accuracies ranging from 50-80% (Madigan et al., 2005; Luyckx and Daelemans, 2008). For about 800 authors, Hitschler et al. (2017) achieve 13% accuracy with a CNN, and Shrestha et al. (2017) also utilize a CNN to attribute the correct author out of 1,000 candidates with an accuracy of 36%.

Koppel et al. (2006, 2011) conducted two experiments on blogs with 10,000 authors. First, they achieve about 35% by using inverse-document-frequencies of stylistic features, represented in a vector space and compared using cosine similarity. In a second study, aiming for precision rather than recall (i.e., to rather output *don't know* than to guess), they use *space-free character 4-grams* with cosine similarity, and enhance their approach by iteratively evaluating randomized subsets of features. By doing so, they report a precision of 93% for the cases an answer is given.

Finally, the most comprehensive study with respect to number of candidate authors has been conducted by Narayanan et al. (2012), who evaluate different features with several machine learning techniques on a dataset consisting of 100,000 authors of blogs. In their study, the main focus is laid on security concerns, i.e., that the correct author can be identified in an attack. The authors show that a combination of a simple nearest neighbor approach with a regularized least squares classifier is able to detect the correct author of a blog in 20% of the cases and that the correct author is in the top-20 ranked candidates in 35% of the cases. Moreover and along the lines of Luyckx and Daelemans (2011) or Eder (2010), it is shown that the size of available training/test texts substantially influences the performance.

As the studies of (Koppel et al., 2011) and (Narayanan et al., 2012) are the only ones targeting authors in the large scale, we will also use these studies as references throughout this paper (in terms of their methodology and reported results). Nevertheless, a direct comparison is difficult as they either target different aims and/or the underlying datasets are not or only partly available. In contrast to these studies, we propose a novel two-step method in this paper and provide comprehensive large scale studies alongside, which can easily be reproduced in both methods as well as data used.

3 The SE-179 Dataset Collection

For the task of authorship attribution, a suitable dataset has to consist of realistic documents where the authorship of each document can undoubtedly be attributed to a single author. In the case of single-domain or single-topic analyses, it has to additionally be assured that all candidate authors write about the same topics—such that they cannot be exposed by simply looking at specific topic-related content words. Along the lines of Kestemont et al. (2018), who showed that data from Q&A forums can successfully be employed to analyze the writing style, we also used the same Q&A platform, namely *StackExchange*[3], to create our dataset.

The StackExchange network consists of several sites where people answer questions related to specific topics (*Stackoverflow* being the most popular site). In contrast to Kestemont et al. (2018) where only selected posts of selected StackExchange sites were crawled, we use the provided data dump[4] containing all questions and answers of all sites. Because the posts for each site are related to a single topic (e.g., photography), it allows us to create individual datasets from each site. Thereby the procedure for creating a dataset from a site was as follows:

[3] https://stackexchange.com

[4] provided directly by StackExchange at https://archive.org/details/stackexchange

(1.) We collected all questions and answers by all users participating in the site. (2.) We removed all posts that were edited by a person different from the original author (in the StackExchange network, basically everyone can edit anyone's posts). (3.) We cleaned each post, i.e., we removed code snippets, block quotes, bullet lists, embedded images and replaced links with `$URL$`. We then dismissed all posts containing less than ten tokens after cleaning. (4.) We combined all remaining posts of each user into a single document and removed all users with less than 500 tokens. Subsequently, we divided each document into a training and a test document. Thereby we assured that each training and test document contains at least 500 tokens, and in case this was not possible (because there were less than 1,000 tokens available), we only kept the training document to increase the number of candidates. I.e., there exist several training documents where there exists no corresponding test document. Note that it is a common procedure to fix training and test documents in order to ensure reproducibility (Stamatatos et al., 2018).

Consequently, we created a balanced, single-topic dataset from each site, containing different numbers of authors depending on the size of the community of the respective site. Table 1a shows the statistics of the resulting 179 datasets[5], including the average tokens per document (avg t/d) as well as the ratio between number of training to test documents (ttr). We consequently call the overall dataset collection *SE-179*. With respect to languages throughout the collection, the predominant one is English, but also individual problems in different languages are present[6].

For our study, the datasets containing many authors as listed in Table 1b are of high interest, nevertheless we conducted our experiments also on all other datasets as is detailed in Section 4.2. By doing so we can avoid potential biases towards specific datasets. As we are concerned about reproducibility, we make the SE-179 collection publicly available and encourage other researchers to utilize it according to their needs.[7]

[5]We didn't process the *Stackoverflow* site due to computational limitations with respect to its size.

[6]I.e., one for each Chinese, Esperanto, French, German, Italian, Japanese, Korean, Polish, Portuguese, Russian, Spanish and Ukrainian.

[7]The dataset is available at https://doi.org/10.5281/zenodo.3441861.

authors	datasets	train avg t/d	test avg t/d	ttr
≤ 10	3	757	1143	45%
11–100	31	740	569	47%
101–250	34	698	569	46%
251–500	31	703	577	46%
501–1,000	33	684	574	45%
1,001–5,000	38	669	566	45%
5,001–10,000	7	677	572	44%
$> 10{,}000$	2	613	552	40%

(a) General statistics

site origin	authors	site origin	authors
Superuser.com	19,272	Softwareengineering	6,351
Serverfault.com	16,450	Electronics	6,119
Askubuntu.com	9,830	Unix	5,507
Physics	7,418	Stats	5,307
Mathoverflow.net	6,361	Wordpress	3,898

(b) Top 10 large scale datasets.

Table 1: Statistics of the SE-179 dataset collection including average tokens per document (avg t/d) and the ratio between number of training and test documents (ttr).

4 Reducing Candidate Authors

In this section, we outline our approach of effectively reducing the number of candidate authors by utilizing document embeddings. After describing the applied technique in Section 4.1 as well as the reference implementations used, we show the results in Section 4.2.

4.1 Methods

Document Embeddings

Based on the promising results reported by Posadas-Durán et al. (2017), we also utilize document embeddings with doc2vec (Le and Mikolov, 2014). Considering the two possible representation techniques provided by doc2vec, i.e., distributed memory (DM) and distributed bag of words (DBOW), we evaluated the three basic models (i) DM using concatenation (DM/concat), (ii) DM using average (DM/avg), (iii) DBOW as well as the two combinations (iv) DBOW+DM/concat and (v) DBOW+DM/avg. For each model, we evaluated vector sizes (dimensions) of $d = \{100, 200, 300\}$ (or the double in case of the combined models) and relied on the default/optimal settings found by Posadas-Durán et al. (2017) for the specific model parameters. With respect to the textual input, we at first tokenize the text and then experiment with the following settings to compute the embeddings:

– *type*: we either provide the text as is (*unigram*) or we compute *bigrams* of the words

– *stem*: decides whether stemming should be applied or not

– *windowing*: if a window length (w_l) is set, we traverse the document using a sliding window containing w_l tokens and thereby create new "documents" for each author. The window step w_s defines the number of tokens the window is shifted after each iteration. Additionally, we compute models from the original documents without windowing.

For the final reduction of candidate authors according to a given test document, the procedure is as follows: (1.) According to the previously described settings, we learn models from all available training documents of all authors. During this step, all documents are assigned a vector of dimension d (the model dimension), which form an according vector space. In case windowing is used, each author is represented by several vectors (one per window). (2.) For the given test document, we apply the same preprocessing steps (i.e., input type, stemming and windowing) and make use of the functionality provided by doc2vec to estimate a vector for a document that was not seen during learning. (3.) Similar to Koppel et al. (2011) we then compare the test document's vector with all document vectors in the vector space by computing the cosine similarity. Ordering by this similarity and using the top-k authors finally allows reducing the set of candidates to an arbitrary extent. In case windowing is used, i.e., when there are multiple documents by each author, we use the average of the similarities of all the author's document vectors.

Reference Implementations

To compare the proposed approach, we reimplemented the approach described by Koppel et al. (2011). Specifically, in this approach so-called *space-free* character 4-grams are computed for each document and their normalized frequencies form the basis for a vector space. By repeatedly (k_1 times) selecting k_2% of the feature set randomly, cosine similarity is used to compare the documents. In our reimplementation we used the optimal values as reported, i.e., $k_1 = 100$ and $k_2 = 40\%$. As an additional reference, we used regular n-grams instead of the space-free variants.

authors	**model**	d	stem	w_l	w_s	type
≤ 10	DM/avg	200	yes	–	–	unigram
11–100	DM/avg	100	yes	–	–	unigram
101–250	DM/concat	100	yes	300	50	unigram
251–500	DM/concat	100	yes	300	50	unigram
501–1,000	DM/concat	100	yes	300	50	unigram
1,001–5,000	DM/concat	200	yes	300	50	unigram
5,001–10,000	DM/concat	100	yes	–	–	unigram
$> 10{,}000$	DM/concat	100	yes	–	–	unigram

Table 2: Best doc2vec models with respect to number of authors.

4.2 Estimating Best Reduction Models

Contrary to Narayanan et al. (2012) we find it more suitable to not test whether the correct author is in the top-k results, but to evaluate how often s/he is in the result set after reducing by percentage (e.g., eliminating 90% of the candidates). This makes especially sense as we are dealing with 179 different datasets of different sizes, where a comparison of the top-k results with a fixed k is not meaningful (e.g., it makes a huge difference if the correct author is in the top-5 in a dataset containing 20 authors or in one containing 16,000 authors). Thus, we experimented with the reduction rates 10-90%, 95%, and 99%, and measured the hit rate, i.e., the percentage of how often the correct author is still in the reduced candidate set.

In a first preliminary step, we aimed to find the best models with respect to reduction rate and candidate author size[8]. We evaluated on all 179 datasets using the respective training documents for learning and the test documents for testing. For the larger datasets, we tested on 1,000 randomly selected test documents (as has been done by Koppel et al. (2011)).

After conducting the experiment, we found that the reduction rate doesn't make any difference with respect to the model type and that the best performing models only depend on the number of authors. Table 2 shows the best settings for different number of authors, computed by using the average of all corresponding datasets and regardless of the reduction rate.[9] It can be seen that stemmed unigrams work best in all cases and that windowing is not the preferred option when looking at large (and small) candidate sizes.

Using the best models found (depending on the number of candidate authors) we evaluated their

[8] E.g., what is the best doc2vec-model for reducing an 8,000 author dataset by 70%?

[9] Note that we cannot provide single hit rates for each configuration, as they significantly depend on the reduction rate.

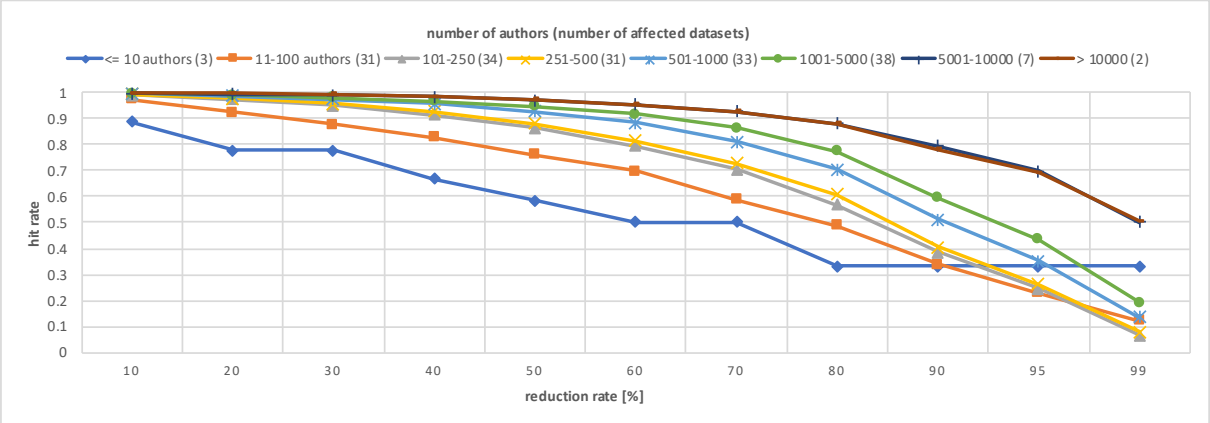
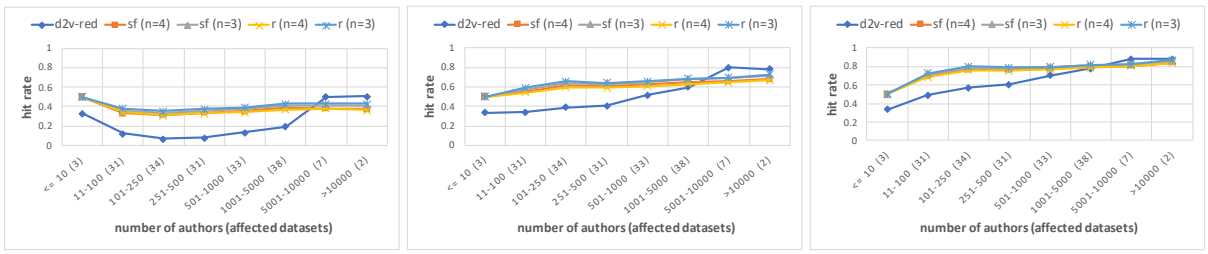


Figure 1: Hit rates for the doc2vec reduction models averaged over all datasets with respect to reduction rate and candidate author size.



(a) reduction rate = 99%

(b) reduction rate = 90%

(c) reduction rate = 80%

Figure 2: Comparison of the doc2vec reduction (d2v-red) with the method proposed by Koppel et al. (2011) using space-free (sf) and regular (r) n-grams. The y-axis shows the average hit rates over all datasets for different reduction rates, grouped by number of candidate authors.

performance with respect to the reduction rate. That is, we reduced the number of candidates by the respective percentage and measured—in terms of hit rate—if the correct author is still in the remaining set. As expected and can be seen in Figure 1, the performance decreases as the reduction rate increases. In general, the more candidate authors, the better the model is able to filter out irrelevant ones: E.g., the datasets having more than 5,000 authors could be reduced by 50% with a hit rate of 0.97, and by 80% with a hit rate of 0.88. When reducing these datasets by 99%, a hit rate of approximately 0.5 remains.

In a further experiment, we compared the reduction results to the reference systems described in Section 4.1, i.e., with regular and space-free n-grams as proposed by Koppel et al. (2011), Figure 2 exemplarily shows the average results over all datasets, grouped by the number of candidate authors for the reduction rates 99%, 90% and 80%, respectively. Regardless of the individual reduction rate, the doc2vec model is inferior to the other models for datasets having less than 5,000 authors, but can significantly[10] exceed them when more authors are involved. For example, when reducing candidates by 90% in a 5,000+ candidate author setting, it is able to keep the correct author with a hit rate of 0.69 in average, whereas the best other model (regular 3-grams) achieves a hit rate of 0.60. Although the superiority of our doc2vec model decreases with lower reduction rates, it is still better than the other models for all reduction rates in scenarios having more than 5,000 candidate authors, as is shown in Table 3.

5 Attribution on Reduced Candidates

In the previous section, we have shown that the number of candidate authors in large authorship attribution problems can effectively be reduced by compiling a document embedding model based on word unigrams. As a follow-up, we wanted to assess the influence of this reduction technique for state-of-the-art authorship attribution methods. The basic idea is to apply a two-step attribution by

[10] We computed a McNemar's test (Dietterich, 1998) and interpreted $p < 0.05$ as significant.

red.	d2v-red	sf (n=4)	sf (n=3)	r (n=4)	r (n=3)
10%	**0.998**	0.994	0.993	0.994	0.993
20%	**0.995**	0.988	0.986	0.988	0.986
30%	**0.991**	0.982	0.978	0.981	0.979
40%	**0.984**	0.973	0.969	0.973	0.971
50%	**0.972**	0.958	0.954	0.958	0.959
60%	**0.954**	0.931	0.932	0.937	0.938
70%	**0.928**	0.890	0.896	0.895	0.904
80%	**0.881**	0.812	0.827	0.811	0.834
90%	**0.792**	0.659	0.695	0.653	0.700
95%	**0.697**	0.551	0.585	0.538	0.593
99%	**0.501**	0.377	0.412	0.376	0.431

Table 3: Comparison of the proposed doc2vec reduction (d2v-red) with the method proposed by Koppel et al. (2011) using space-free (sf) and regular (r) n-grams. The table shows the average hit rates for the respective reduction rates (red.) over all 9 datasets containing more than 5,000 authors.

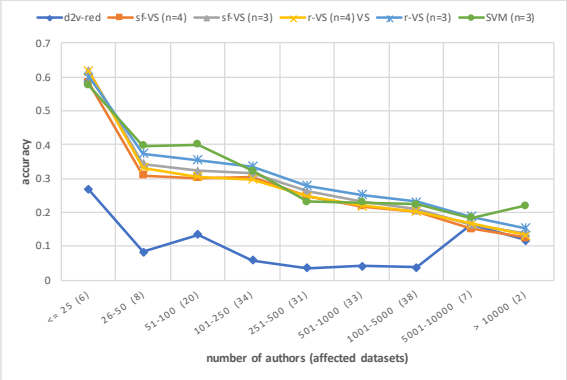


Figure 3: Accuracies for direct authorship attribution averaged over different candidate author sizes. *sf-VS* and *r-VS* represent space-free and regular n-grams, respectively, represented in a vector space as proposed by Koppel et al. (2011), and *SVM* refers to the reference implementation provided by PAN-2018.

transforming large scale problems to normal-scale problems: (1.) reduce the number of candidate authors, (2.) apply regular authorship attribution approaches for the remaining candidates.

5.1 Direct Attribution Baseline

In a first step, we created a baseline by computing the accuracies achieved for direct authorship attribution, i.e., for finding the correct author without any reduction. For this, we utilized the proposed reduction technique, but reduced to exactly one author instead of a set of authors. Similar to Section 4.1, we again utilized the approach of Koppel et al. (2011) with space-free and regular character 3-/4-grams in combination with a vector space and cosine similarity. As an additional reference for comparison, we made use of the reference implementation provided for the author identification task at the PAN 2018 event (Kestemont et al., 2018). It computes character 3-grams and makes classifications using a standard SVM, yet achieving competitive results by applying grid search (Murauer et al., 2018). The results averaged over candidate author sizes are presented in Figure 3, revealing that doc2vec is very imprecise for direct authorship attribution in non-large cases. The other approaches generally perform similarly, except for the largest datasets where the SVM achieved the best results (0.21 for the datasets with more than 10,000 authors).

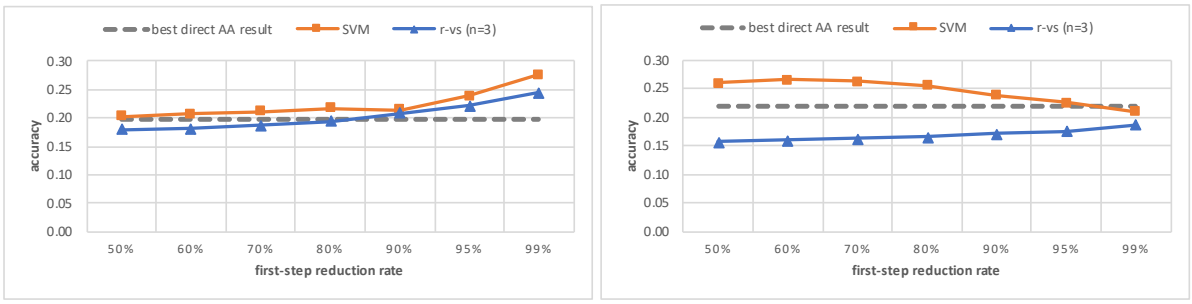
5.2 Two-Step Attribution

To measure the influence of the reduction proposed in Section 4, we conducted an experiment on the largest datasets by at first reducing the number of candidates, and subsequently performing regular authorship attribution. Considering the best results of direct attribution as presented previously, we evaluated the regular 3-gram vector space approach and the SVM implementation of PAN[11]. For each approach, we at first reduced the authors by the respective reduction rate, applied the two approaches and compared it to the best result achieved by direct attribution.

Figure 4 depicts the results for the 5,000-10,000 author datasets and for those having more than 10,000 authors, respectively. It can be seen that in general the accuracy–especially that of the SVM–can be improved, nevertheless, the best first-step reduction rate depends on the problem size: For datasets up to 10,000 candidates, the best option is to reduce the number of authors by 99% before performing attribution. On the contrary, the best accuracy for problems with more than 10,000 candidates could be achieved by using a reduction rate of 60%.

As stated initially in the paper, the evaluation results of authorship attribution techniques is highly dependent on the dataset (Luyckx and Daelemans, 2011; Potthast et al., 2016), and while our datasets within the SE-179 collection are highly heterogeneous with respect to topics and author sizes and also languages, they still belong to the genre of question-answering platforms. We therefore aimed to evaluate the dataset used by Koppel et al. (2011) to gain additional insight into the perfor-

[11]Note that for each test document a corresponding SVM has to be trained on the remaining candidate authors after reduction.



(a) 5k-10k authors datasets

(b) > 10k authors datasets

Figure 4: Two-step attribution with preliminary candidate reduction using doc2vec and cosine similarity.

mance on a different genre, i.e., blogs. Unfortunately, the data is only available in a raw form, i.e., we had to reconstruct the dataset as to the best of our knowledge incorporating the facts given by the authors. Consequently, we ended up with blog entries from 10,000 authors, each containing about 2,000 words, whereby the first 1,500 words were used for training and the last 500 for testing.

Considering the superiority of the SVM with character 3-grams in the previous experiment, we compared the performance of the SVM on all datasets of the SE-179 collection with more than 5,000 authors with the performance on the recreated blog dataset. Figure 5 shows the relative improvements of our proposed two-step attribution compared to the best direct attribution results, which are very similar for both the SE-179 datasets and the blog dataset, i.e., 0.202 and 0.204, respectively. As can be seen, a preliminary reduction of candidates substantially improves the performance, especially for the blog dataset for which the accuracy could be increased by more than 10%.

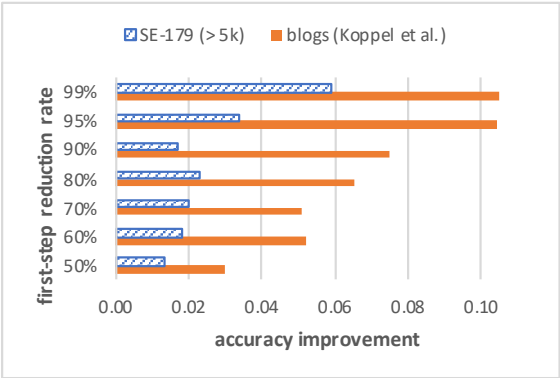


Figure 5: Relative performance improvements of the PAN-2018-SVM baseline implementation on SE-179 datasets with more than 5,000 authors and the reproduced blogs dataset (Koppel et al., 2011). The chart shows the improvements in accuracy that could be gained by applying preliminary candidate reduction.

6 Conclusion and Future Work

In this paper, we tackled the problem of large scale authorship attribution incorporating thousands of authors by first filtering candidate authors before the actual classification step. Extensive evaluations on a novel, publicly available dataset collection reveal that document embeddings in combination with cosine similarity are able to effectively reduce the number of candidate authors for large scale problems. We also outlined that a preliminary reduction increases the overall attribution accuracy in such cases.

As for future work, several open issues should be addressed. In this study, we relied on related work that suggests document embeddings, nevertheless other embedding techniques like *word2vec* (Mikolov et al., 2013), *fastText* (Joulin et al., 2016) or *GloVe* (Pennington et al., 2014) could be evaluated. Moreover, for the computation of similarities between document vectors, we relied on cosine similarity, whereas several other metrics should be evaluated. In the case of authorship attribution, we similarly utilized a common, established technique (SVM). As the reduction of problem sizes additionally enables the utilization of resource-intensive algorithms, more experiments are needed in that direction, especially using deep learning techniques. Finally, it would be worth investigating how this approach performs on cross-domain/-topic scenarios and other text genres like short messages or other social media contents.

References

Hayri Volkan Agun and Ozgur Yilmazel. 2017. Document embedding approach for efficient authorship attribution. In *Knowledge Engineering and Applications (ICKEA), 2017 2nd International Conference on*, pages 194–198. IEEE.

Shlomo Argamon, Marin Šarić, and Sterling S Stein. 2003. Style mining of electronic messages for multiple authorship discrimination: First results. In *Proceedings of the 9th International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 475–480, Washington, DC, USA. ACM.

José Eleandro Custódio and Ivandré Paraboni. 2018. Each-usp ensemble cross-domain authorship attribution. *Working Notes Papers of the CLEF*.

Thomas G Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923.

Maciej Eder. 2010. Does size matter? authorship attribution, small samples, big problem. In *Proceedings of the Digital Humanities Conference*, London, UK. ALLC.

Jack Grieve. 2007. Quantitative authorship attribution: An evaluation of techniques. *Literary and Linguistic Computing*, 22(3):251–270.

Julian Hitschler, Esther van den Berg, and Ines Rehbein. 2017. Authorship attribution with convolutional neural networks and pos-eliding. In *Proceedings of the Workshop on Stylistic Variation*, pages 53–58.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Patrick Juola. 2012. An overview of the traditional authorship attribution subtask. In *Notebook Papers of the 7th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Rome, Italy.

Patrick Juola and Efstathios Stamatatos. 2013. Overview of the author identification task at pan 2013. In *Notebook Papers of the 9th Evaluation Lab on Uncovering Plagiarism, Authorship and Social Software Misuse (PAN)*, Valencia, Spain.

Mike Kestemont, Michael Tschuggnall, Efstathios Stamatatos, Walter Daelemans, Günther Specht, Benno Stein, and Martin Potthast. 2018. Overview of the author identification task at pan-2018: cross-domain authorship attribution and style change detection. In *Working Notes Papers of the CLEF 2018 Evaluation Labs. Avignon, France, September 10-14, 2018/Cappellato, Linda [edit.]; et al.*, pages 1–25.

Moshe Koppel and Jonathan Schler. 2003. Exploiting stylistic idiosyncrasies for authorship attribution. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, volume 69, pages 72–80, Acapulco, Mexico.

Moshe Koppel, Jonathan Schler, and Shlomo Argamon. 2011. Authorship attribution in the wild. *Language Resources and Evaluation*, 45(1):83–94.

Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Eran Messeri. 2006. Authorship attribution with thousands of candidate authors. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–660. ACM.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Kim Luyckx and Walter Daelemans. 2008. Authorship attribution and verification with many authors and limited data. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 513–520. Association for Computational Linguistics.

Kim Luyckx and Walter Daelemans. 2011. The effect of author set size and data size in authorship attribution. *Literary and linguistic Computing*, 26(1):35–55.

David Madigan, Alexander Genkin, David D Lewis, Shlomo Argamon, Dmitriy Fradkin, and Li Ye. 2005. Author identification on the large scale. In *Proc. of the Meeting of the Classification Society of North America*, volume 13.

Yuval Marton, Ning Wu, and Lisa Hellerstein. 2005. On compression-based text classification. In *Advances in Information Retrieval*, pages 300–314. Springer.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

F. Mosteller and D. Wallace. 1964. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley.

Benjamin Murauer, Michael Tschuggnall, and Günther Specht. 2018. Dynamic parameter search for cross-domain authorship attribution. *Working Notes of CLEF*.

Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. 2012. On the feasibility of internet-scale author identification. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 300–314. IEEE.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Juan-Pablo Posadas-Durán, Helena Gómez-Adorno, Grigori Sidorov, Ildar Batyrshin, David Pinto, and Liliana Chanona-Hernández. 2017. Application of the distributed document representation in the authorship attribution task for small corpora. *Soft Computing*, 21(3):627–639.

Martin Potthast, Sarah Braun, Tolga Buz, Fabian Duffhauss, Florian Friedrich, Jörg Marvin Gülzow, Jakob Köhler, Winfried Lötzsch, Fabian Müller, Maike Elisa Müller, et al. 2016. Who wrote the web? revisiting influential author identification research applicable to information retrieval. In *European Conference on Information Retrieval*, pages 393–407. Springer.

Dylan Rhodes. 2015. Author attribution with cnns. *Avaiable online: https://www. semanticscholar. org/paper/Author-Attribution-with-Cnn-s-Rhodes/0a904f9d6b47dfc574f681f4d3b41bd840871b6f/pdf (accessed on 22 August 2016)*.

Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 93–102.

Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. 2013. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *Computing Surveys*, 34(1):1–47.

Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. 2017. Convolutional neural networks for authorship attribution of short texts. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 669–674.

Efstathios Stamatatos. 2009. A Survey of Modern Authorship Attribution Methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556.

Efstathios Stamatatos. 2013. On the robustness of authorship attribution based on character n-gram features. *Journal of Law and Policy*, 21(2):421–439.

Efstathios Stamatatos. 2017. Authorship attribution using text distortion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1138–1149.

Efstathios Stamatatos, Francisco Rangel, Michael Tschuggnall, Benno Stein, Mike Kestemont, Paolo Rosso, and Martin Potthast. 2018. Overview of pan 2018. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 267–285. Springer.

Benno Stein, Nedim Lipka, and Peter Prettenhofer. 2011. Intrinsic plagiarism analysis. *Language Resources and Evaluation*, 45(1):63–82.

Michael Tschuggnall and Günther Specht. 2014. Enhancing authorship attribution by utilizing syntax tree profiles. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL), volume 2: Short Papers*, pages 195–199, Gothenburg, Sweden. Association for Computational Linguistics.

Fiona J Tweedie and R Harald Baayen. 1998. How variable may a constant be? measures of lexical richness in perspective. *Computers and the Humanities*, 32(5):323–352.

Ying Zhao and Justin Zobel. 2005. Effective and scalable authorship attribution using function words. In *Proceedings of the Information Retrieval Technology: Second Asia Information Retrieval Symposium (AIRS)*, pages 174–189. Springer.

Ying Zhao and Justin Zobel. 2007. Searching with style: Authorship attribution in classic literature. In *Proceedings of the thirtieth Australasian conference on Computer science-Volume 62*, pages 59–68. Australian Computer Society, Inc.

Rong Zheng, Jiexun Li, Hsinchun Chen, and Zan Huang. 2006. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American Society for Information Science and Technology*, 57(3):378–393.

Variational Semi-supervised Aspect-term Sentiment Analysis via Transformer

Xingyi Cheng*, **Weidi Xu** *, **Taifeng Wang, Wei Chu,**
Weipeng Huang, Kunlong Chen and **Junfeng Hu**
Ant Financial Services Group
fanyin.cxy@antfin.com

Abstract

Aspect-term sentiment analysis (ATSA) is a long-standing challenge in natural language processing. It requires fine-grained semantical reasoning about a target entity appeared in the text. As manual annotation over the aspects is laborious and time-consuming, the amount of labeled data is limited for supervised learning. This paper proposes a semi-supervised method for the ATSA problem by using the Variational Autoencoder based on Transformer. The model learns the latent distribution via variational inference. By disentangling the latent representation into the aspect-specific sentiment and the lexical context, our method induces the underlying sentiment prediction for the unlabeled data, which then benefits the ATSA classifier. Our method is classifier-agnostic, i.e., the classifier is an independent module and various supervised models can be integrated. Experimental results are obtained on the SemEval 2014 task 4 and show that our method is effective with different five specific classifiers and outperforms these models by a significant margin.

1 Introduction

Aspect based sentiment analysis (ABSA) has two sub-tasks, namely aspect-term sentiment analysis (ATSA) and aspect-category sentiment analysis (ACSA). ACSA is to infer the sentiment polarity with regard to the predefined categories, e.g., the aspect $food, price, ambience$. On the other hand, ATSA aims at classifying the sentiment polarity of a given aspect word or phrase in the text. For example, given a review about a restaurant "*the* $[pizza]_{aspect}$ *is the best if you like thin crusted pizza, however, the* $[service]_{aspect}$ *is awful.*", the sentiment implications with regard to "*pizza*" and "*service*" are contrary. For the aspect "*pizza*", the sentiment polarity is "*positive*" while "*negative*" for the aspect "*service*". In contrast to document-level sentiment analysis, ATSA requires more fine-grained reasoning about the textual context. The task is worthy of investigation as it can obtain the attitude with regard to a specific entity which we are interested in. The task is widely applicated in analyzing the comments, such as opinion generation. Recently, many attempts (Tang et al., 2016b; Pan and Wang, 2018; Liu et al., 2018; Xue and Li, 2018; Li et al., 2018a) focus on supervised learning and pay much attention to the interaction between the aspect and the context. However, the amount of labeled data is quite limited as the annotation about the aspects is laborious. Currently available data sets, e.g. SemEval, only has around 2K unique sentences and 3K sentence-aspect pairs, which is insufficient to fully exploit the power of the deep models. Fortunately, a large amount of unlabeled data is available for free and can be accessed easily from the websites. It will be of great significance if numerous unlabeled samples can be utilized to further facilitate the supervised ATSA classifier. Therefore, the semi-supervised ATSA is a promising research topic.

In ATSA, achieving the sentiment of the aspect-term is semantically complicated and it is non-trivial for a model to capture sentimental similarity of the aspects, which causes the difficulties for semi-supervised learning. In this paper, we proposed a classifier-agnostic framework which named Aspect-term Semi-supervised Variational Autoencoder (Kingma and Welling, 2014) based on Transformer (ASVAET). The variational autoencoder offers the flexibility to customize the model structure. In other words, the proposed framework is compatible with other supervised neural networks to boost their performance. Our proposed model learns the latent representation

*: Equal Contribution

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 961–969
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

of the input data and disentangles the representations into two independent parts, i.e., the aspect-term sentiment and the representation of the lexical context. By regarding the aspect sentiment polarity of the unlabeled data as the discrete latent variable, the model implicitly induces the sentiment polarity via the variational inference. Specifically, the representation of the lexical context is extracted by the encoder and the aspect-term sentiment polarity is inferred from the specific ATSA classifier. The decoder takes these two representations as inputs and reconstructs the original sentence by two unidirectional language models. In contrast to the conventional auto-regressive models, the latent representations have their specific meanings and are obtained from the encoder and the classifier to the input examples. Therefore, it is also possible to condition the sentence generation on the sentiment and lexical information w.r.t. a certain target entity. In addition, by separating the representation of the input sentence, the classifier becomes an independent module in our framework, which endows the method with the ability to integrate different classifiers. The method is presented in detail in Sec. 3.

Experimental results are obtained on the two classical datasets from SemEval 2014 task 4 (Pontiki et al., 2014). Five recent available models are implemented as the classifier in ASVAET. Our method is able to utilize the unlabeled data and consistently improve the performance against the supervised models. Compared with other semi-supervised methods, i.e., in-domain word embedding pre-training and self-training, the proposed method also demonstrates better performance. We also evaluate the effectiveness of labeled data and sharing embeddings, and show that the structure can provide the separation between lexical context and sentiment polarity in the latent space.

2 Related Work

Sentiment analysis is a traditional research hotspot in the NLP field (Wang and Manning, 2012). Rather than obtaining the sentimental inclination of the entire text, ATSA instead aims to extract the sentimental expression w.r.t. a target entity. With the release of online completions, abundant methods were proposed to explore the limits of current models. Tang et al. (Tang et al., 2016a) proposed to make use of bidirectional Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to encode the sentence from the left and right to the aspect-term. This model primarily verifies the effectiveness of deep models for ABSA Tang et al. (Tang et al., 2016b) then put forward a neural reasoning model in analogy to the memory network to perform the reasoning in many steps. There are also many other works dedicating to solve this task (Pan and Wang, 2018; Liu et al., 2018; Zhang and Liu, 2017).

Another related topic is semi-supervised learning for the text classification. Recently, Data augmentation methods (Xie et al., 2019; Berthelot et al., 2019) achieve a greate success on low-resource datasets. Moreover, A simple but efficient method is to use pre-trained modules, e.g., initializing the word embedding or bottom layers with pre-training. Word embedding technique has been wildly used in NLP models, e.g., Glove (Pennington et al., 2014) and ELMo (Peters et al., 2018). Recently, Bidirectional Encoder Representations from Transformer (BERT) (Devlin et al., 2018) replaces the embedding layer to context-dependent layer with the pre-trained bidirectional language model to capture the contextual representation. BERT is complementary to the encoder of the proposed method. To keep our framework neat, these pre-training investigations are not conducted in this paper.

VAE-based semi-supervised methods, on the other hand, are able to cooperate with various kinds of classifiers. VAE has been applied in many semi-supervised NLP tasks, ranging from text classification (Xu et al., 2017), relation extraction (Marcheggiani and Titov, 2016) to sequence tagging (Chen et al., 2018). Different from text classification where sentiment polarity is related to an entire sentence, ATSA just interested in related information of a given aspect-term. To circumvent this problem, a novel structure is proposed.

3 Method Description

In this section, the problem definition is provided and then the model framework is presented in detail.

The ATSA task aims to classify a data sample with input sentence $\mathbf{x} = \{x_1, ..., x_n\}$ and corresponding aspect [1] $\mathbf{a} = \{a_1, ..., a_m\}$, where $\mathbf{a}$ is a subsequence of $\mathbf{x}$, into a sentiment polarity y, where $y \in \{P, O, N\}$. P, O, N denotes

[1] If an input sentence has n aspect-terms, then n data samples are generated.

"positive", "neutral", "negative". For the semi-supervised ATSA, we consider the following scenario. Given a dataset consisting of labeled samples $\mathbf{S}_l$ and unlabeled samples $\mathbf{S}_u$, where the $\mathbf{S}_l = \{(\mathbf{x}_l^{(i)}, \mathbf{a}_l^{(i)}, y_l^{(i)})\}_{i=1}^{N_l}$ and $\mathbf{S}_u = \{(\mathbf{x}_u^{(i)}, \mathbf{a}_u^{(i)}\}_{i=1}^{N_l}$, the goal is to utilize $\mathbf{S}_u$ to improve the classification performance over the supervised model using $\mathbf{S}_l$ only.

The architecture is depicted in Fig. 1. The method consists of three main components, i.e., the classifier, the encoder, and the decoder. The classifier can be any differentiable supervised ATSA model, which takes $\mathbf{x}$ and $\mathbf{a}$ as input, and outputs the prediction about y. The encoder transform the data into a latent space that is independent of the label y. And the decoder combines the outputs from the classifier and the encoder to reconstruct the input sentence. For the labeled data, the classifier and the autoencoder are trained with the given label y. For the unlabeled data, the y is regarded as the latent discrete variable and it is induced by maximizing the generative probability. As the classifier can be implemented by various models, the description of the classifier will be omitted. We present a autoencoder structure based on Transformer (Vaswani et al., 2017). In the following, the objective functions are clarified, followed by the model description.

3.1 Variational Inference

Using generative models is a common approach for semi-supervised learning, which tries to extract the information from the unlabeled data by modeling the data distribution. In VAE, the data distribution is modeled by optimizing the evidence lower bound (ELBO) of data log-likelihood, which leads to two objectives for labeled data and unlabeled data respectively. For the labeled data, VAE maximizes the ELBO of $p(\mathbf{x}, y|\mathbf{a})$. For the unlabeled data, it optimizes the ELBO of $p(\mathbf{x}|\mathbf{a})$, where the y is latent and integrated. Specifically, the dependency between variables is illustrated in Fig. 2. The ELBO of $\log p(\mathbf{x}, y|\mathbf{a})$ can be given as follows:

$$\begin{aligned}\log p_\theta(\mathbf{x}, y|\mathbf{a}) \geq\ & \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{a},y)}[\log p_\theta(\mathbf{x}|y,\mathbf{a},\mathbf{z})] \\ & - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{a},y)||p_\theta(\mathbf{z})) \\ & + \log p_\theta(y) \\ =\ & \mathcal{L}(\mathbf{x},\mathbf{a},y)\,,\end{aligned} \tag{1}$$

where $\mathbf{z}$ is the latent variable which represents lexical information over the sentence and D_{KL} is the KullbackLeibler divergence.

In terms of the unlabeled data, the ELBO of $\log p(\mathbf{x}|\mathbf{a})$ can be extended from Eq. 1.

$$\begin{aligned}\log p_\theta(\mathbf{x}|\mathbf{a}) \geq & \sum_y q_\phi(y|\mathbf{x},\mathbf{a})(\mathcal{L}(\mathbf{x},\mathbf{a},y)) \\ & + \mathcal{H}(q_\phi(y|\mathbf{x},\mathbf{a})) \\ = & \mathcal{U}(\mathbf{x},\mathbf{a})\,,\end{aligned} \tag{2}$$

where $\mathcal{H}$ is the entropy function and $q_\phi(y|\mathbf{x},\mathbf{a})$ is the classification function.

And $q_\phi(y|\mathbf{x},\mathbf{a})$ can also be trained in the supervised manner using the labeled data. Combining the above objectives, the overall objective for the entire data set is:

$$\begin{aligned}J = & \sum_{(\mathbf{x},\mathbf{a},y)\in S_l} -\mathcal{L}(\mathbf{x},\mathbf{a},y) + \sum_{\mathbf{x}\in S_u} -\mathcal{U}(\mathbf{x},\mathbf{a}) \\ & + \gamma \sum_{(\mathbf{x},\mathbf{a},y)\in S_l} -\log q_\phi(y|\mathbf{x},\mathbf{a})\,,\end{aligned} \tag{3}$$

where γ is a hyper-parameter which controls the weight of the additional classification loss.

To implement this objective, three components are required to model the $q_\phi(y|\mathbf{x},\mathbf{a})$, $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{a},y)$ and $p_\theta(\mathbf{x}|y,\mathbf{a},\mathbf{z})$ respectively.

3.2 Classifier

Various currently available models can be used as the classifier. For the unlabeled data, the classifier is used to predict the distribution of label y for the decoder, i.e., $y \sim q_\phi(y|\mathbf{x},\mathbf{a})$. The distribution $q_\phi(y|\mathbf{x},\mathbf{a})$ will be tuned during maximizing the objective in Eq. 2. In this work, five classifiers are implemented in ASVAET and they are also used as the supervised baselines for the comparison.

3.3 Transformer Encoder

The encoder plays the role of $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{a},y)$. This module attempts to extract the lexical feature that is independent of the label y when given data sample $(\mathbf{x},\mathbf{a})$. In this way, the $\mathbf{z}$ and $\mathbf{y}$ jointly form the representative vector for the input data.

In our implementation, we use a bidirectional encoder to construct sentences embeddings. It is referred as the Transformer encoder that is actually a sub-graph of the Transformer architecture (Vaswani et al., 2017), the architecture is shown in the left part of the Fig. 2. The encoder employs residual connections around each of the multi-head attention sub-layers, followed by

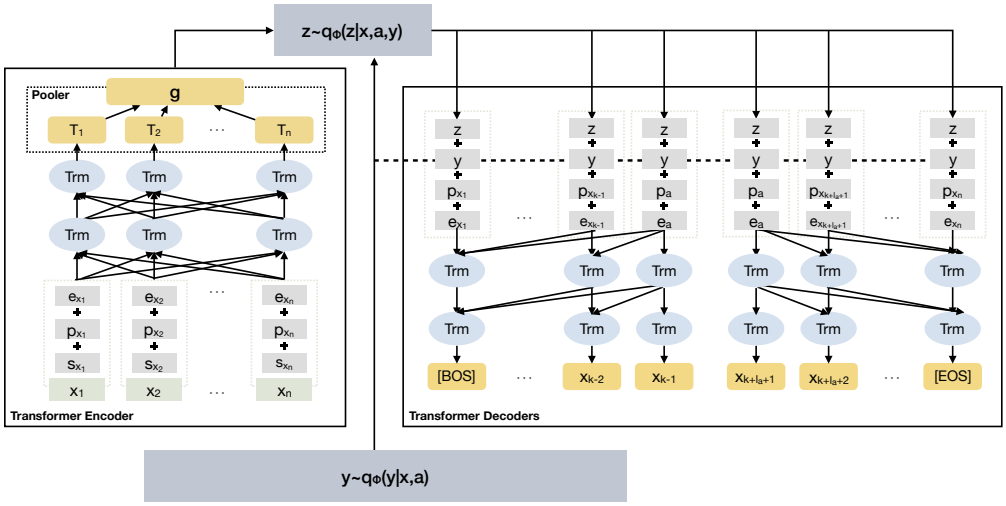


Figure 1: This is the sketch of our model with bidirectional encoder and decoder. Assuming the aspect-term starts at the k-th position in $\mathbf{x}$. **Bottom**: When using unlabeled data, the distribution of $y \sim q_\phi(y|\mathbf{x}, \mathbf{a})$ is provided by the classifier. **Left**: The sequence is encoded by a Transformer block, which receives the summation of three embeddings, i.e., segment (used to distinguish aspect words) $\mathbf{s}_{x_n}$, position $\mathbf{p}_{x_n}$ and word $\mathbf{e}_{x_n}$. The encoding and the label y are used to parameterize the posterior $q_\phi(z|\mathbf{x}, \mathbf{a}, y)$. **Right**: A sample $\mathbf{z}$ from the posterior $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{a}, y)$ and label y are passed to the generative network which estimates the probability $p_\theta(\mathbf{x}|y, \mathbf{a}, \mathbf{z})$ by two unidirectional Transformer decoders. The number of aspect tokens is l_a.

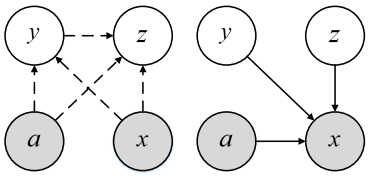


Figure 2: Illustration of ASVAET as a directed graph. **Left**: Dashed lines are used to denote variational approximation $q_\phi(y|\mathbf{x}, \mathbf{a})q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{a}, y)$. **Right**: Solid lines are used to denote generative model $p_\theta(\mathbf{x}|y, \mathbf{a}, \mathbf{z})$.

layer normalization. To capture the aspect-term, we treat the aspect-term and its context differently by segment embeddings. To further emphasize the position of the conditional aspect, the position tag is also included for each token. The position tag indicates the distance from the token to the aspect. And then the position tag is transformed into a vector as defined in (Vaswani et al., 2017), which is added with the word embedding and segment embedding as the input of the Transformer encoder. Let $\mathbf{g}$ denote the output of the Transformer encoder after pooling which simply averaging the hidden states of the aspect-terms (the number of tokens is equal or greater than one) of the last layer, $\mathbf{y}$ is the indicator vector of the polarity. Then the distribution of $\mathbf{z}$ can be given as:

$$\mathbf{z} \sim \mathcal{N}(\mu(\mathbf{x}, y), diag(\sigma^2(\mathbf{x}, y))) ,$$
$$\mu(\mathbf{x}, y) = \tanh(\mathbf{W}_\mu[\mathbf{g} : \mathbf{y}] + \mathbf{b}_\mu) ,$$
$$\sigma(\mathbf{x}, y) = \tanh(\mathbf{W}_\sigma[\mathbf{g} : \mathbf{y}] + \mathbf{b}_\sigma) .$$

The sequences are divided into two parts by using segment embedding, the encoder can be aware of the position and the content of the aspect-term $\mathbf{a}$ by multi-head attention operation in the Transformer encoder. The information from two sides are aggregated into the aspect-term $\mathbf{a}$, and therefore the resulting $\mathbf{z}$ can gather the information related to the aspect.

3.4 Transformer Decoders

The decoder is also a sub-graph of Transformer architechture (Vaswani et al., 2017) which focus on reconstructing original text. The main difference from the Transformer encoder is that the Transformer decoder is unidirectional by modifying the self-attention sub-layer to prevent positions from paying attention to subsequent positions. The textual sequence is well-known to be semantically complex and it is non-trivial for a Transformer decoder to capture the high-level semantics. Here we investigate two questions. How to implement $p_\theta(\mathbf{x}|y, \mathbf{a}, \mathbf{z})$ without losing the information of $\mathbf{a}$ and how to capture the semantic polarity by a sequential model. For the first question, denoting

that $\mathbf{x}$ is composed of three parts $(\mathbf{x}_l, \mathbf{a}, \mathbf{x}_r)$, we use two Transformer decoders to model the left and right content. For the second question, we let each token is generated conditioned on the summation of the variables $\mathbf{z}$ and embedding $\mathbf{y}$.

One way to achieve $p_\theta(\mathbf{x}|y, \mathbf{a}, \mathbf{z})$ is to separate the sequence into two parts, reversing the process in the two unidirectional decoder. For each decoder, the input state is represented by the summation of the four input i.e., the polarity indicator vector $\mathbf{y}$ from the classifier or the labeled dataset, the context vector $\mathbf{z}$ from the encoder, the input token embedding $\mathbf{e}_{x_t}$ and the position embedding $\mathbf{p}_{x_t}$:

$$
\begin{aligned}
\overleftarrow{\mathbf{h}}_t^{trm} &= \overleftarrow{f_{trm}}(\mathbf{e}_{[x_t:a]}, \mathbf{p}_{x_t}, \mathbf{y}, \mathbf{z}), \quad x_t \in [\mathbf{x}_l : \mathbf{a}] \\
p(x_{t-1}|\cdot) &= \text{softmax}(\mathbf{W}_p \overleftarrow{\mathbf{h}}_t^{trm} + b_p)\,, \\
\log p_\theta(\mathbf{x}_l|\mathbf{a}, y, \mathbf{z}) &= \sum_{x_t} \log p(x_t|\cdot), \quad x_t \in \mathbf{x}_l\,, \\
\overrightarrow{\mathbf{h}}_t^{trm} &= \overrightarrow{f_{trm}}(\mathbf{e}_{[a:x_t]}, \mathbf{p}_{x_t}, \mathbf{y}, \mathbf{z}), \quad x_t \in [\mathbf{a} : \mathbf{x}_r] \\
p(x_{t+1}|\cdot) &= \text{softmax}(\mathbf{W}_p \overrightarrow{\mathbf{h}}_t^{trm} + b_p)\,, \\
\log p_\theta(\mathbf{x}_r|\mathbf{a}, y, \mathbf{z}) &= \sum_{x_t} \log p(x_t|\cdot), \quad x_t \in \mathbf{x_r}\,.
\end{aligned}
$$

It is equivalent to generate two sequences using two decoders. When decoding left part (or right part), the aspect will first get processed by the decoder and hence the decoder is aware of the aspect-terms. The position tag is also used in the decoder.

4 Experiments

4.1 Datasets and Preparation

The models are evaluated on two benchmarks: Restaurant (REST) and Laptop (LAPTOP) datasets from the SemEval ATSA challenge (Pontiki et al., 2014). The REST dataset contains the reviews in the restaurant domain, while the LAPTOP dataset contains the reviews of Laptop products. The statistics of these two datasets are listed in Table 1. When processing these two datasets, we follow the same procedures as in another work (Lam et al., 2018). The dataset has a few samples that are labeled as "conflict" and these samples are removed. All tokens in the samples are lowercased without other preprocess, e.g., removing the stop words, symbols or digits.

In terms of the unlabeled data, we obtained samples in the same domain for the REST and LAPTOP datasets. For the REST, the unlabeled samples are obtained from a sentiment analysis competition in Kaggle [2]. The competition consists of 82K training samples and 34K test samples. For the LAPTOP, the unlabeled samples are obtained from the "Six Categories of Amazon Product Reviews" [3], which has 412K samples. The reviews about the laptops are used among six product categories.

		# Positive	# Negative	# Neutral
REST	Train	2159	800	632
	Test	730	195	196
LAPTOP	Train	980	858	454
	Test	340	128	171

Table 1: The statistics of the datasets.

		Avg. Length	Std. Length
REST	Labeled	20.06	10.38
	Unlabeled	22.70	12.38
LAPTOP	Labeled	21.95	11.80
	Unlabeled	29.89	17.33

Table 2: The statistics of the reviews.

The NLTK sentence tokenizer is utilized to extract the sentences from the raw comments. And each sentence is regarded as a sample in our model for both REST and LAPTOP. To obtain the aspects in the unlabeled samples, an open-sourced aspect extractor [4] is pre-trained using labeled data. The resulting test F1 score is 88.42 for the REST and 80.12 for the LAPTOP. Then the unlabeled data is processed by the pre-trained aspect extractor to obtain the aspects. The sentences that have no aspect are removed. And the sentences are filtered with maximal sentence length 80. The statistic of the resulting sentences is given in Table. 2.

4.2 Model Configuration & Classifiers

In the experiments, the model is fixed with a set of universal hyper-parameters. The number of units in the encoder and the decoder is 100 and the latent variable is of size 50 and the number of layers of both Transformer blocks is 2, the number of self-attention heads is 8. The KL weight klw should be carefully tuned to prevent the model from trapping in a local optimum, where $\mathbf{z}$ carries no useful information. In this work, the KL weight is set to be 1e-4. In term of word embedding, the pre-trained GloVe (Pennington et al., 2014) is used as the in-

[2] https://inclass.kaggle.com/c/restaurant-reviews
[3] http://times.cs.uiuc.edu/ wang296/Data/
[4] https://github.com/guillaumegenthial/sequence_tagging

Classifier	Models	REST		LAPTOP	
		Accuracy	Macro-F1	Accuracy	Macro-F1
-	CNN-ASP	77.82 ♮	-	72.46 ♮	-
-	AE-LSTM	76.60 ♮	-	68.90 ♮	-
-	ATAE-LSTM	77.20 ♮	-	68.70 ♮	-
-	GCAE	77.28 (0.32) ♮	-	69.14 (0.32) ♮	-
TC-LSTM	TC-LSTM	77.97 (0.16)	67.55 (0.32)	68.42 (0.56)	62.42 (1.10)
	TC-LSTM (EMB)	77.18 (0.38)	65.97 (0.44)	67.51 (0.72)	60.31 (1.28)
	TC-LSTM (ST)	78.19 (0.36)	67.65 (0.43)	68.47 (0.47)	62.54 (0.74)
	TC-LSTM (ASVAET)	**78.34** (0.18)	**68.41** (0.92)	**70.04** (0.53)	**64.23** (0.71)
MemNet	MemNet	78.68 (0.23)	68.18 (0.58)	70.28 (0.32)	64.38 (0.86)
	MemNet (EMB)	79.47 (0.38)	69.06 (0.21)	72.17 (0.44)	65.06 (0.73)
	MemNet (ST)	78.83 (0.20)	68.92 (0.20)	69.52 (0.36)	64.39 (0.67)
	MemNet (ASVAET)	**80.58** (0.23)	**70.06** (0.53)	**73.21** (0.55)	**65.88** (0.45)
IAN	IAN	79.20 (0.19)	68.71 (0.59)	69.48 (0.52)	62.90 (0.99)
	IAN (EMB)	79.46 (0.38)	69.45 (0.38)	70.89 (0.48)	65.27 (0.34)
	IAN (ST)	79.45 (0.11)	69.36 (0.71)	73.25 (0.81)	68.25 (0.76)
	IAN (ASVAET)	**80.23** (0.17)	**70.32** (1.00)	**74.02** (0.42)	**69.39** (0.75)
BILSTM-ATT-G	BILSTM-ATT-G	79.74 (0.22)	69.16 (0.53)	74.26 (0.35)	69.54 (0.53)
	BILSTM-ATT-G (EMB)	80.27 (0.44)	70.33 (0.51)	73.61 (0.30)	68.25 (0.63)
	BILSTM-ATT-G (ST)	80.54 (0.23)	71.88 (0.19)	74.70 (0.41)	70.31 (0.60)
	BILSTM-ATT-G (ASVAET)	**81.11** (0.34)	**72.19** (0.27)	**75.44** (0.32)	**70.52** (0.33)
TNet-AS	TNet-AS	80.56 (0.23)	71.17 (0.43)	76.75 (0.35)	71.88 (0.35)
	TNet-AS (EMB)	80.96 (0.49)	69.99 (0.87)	76.45 (0.40)	71.52 (0.73)
	TNet-AS (ST)	80.76 (0.23)	71.32 (0.56)	76.88 (0.41)	71.74 (0.63)
	TNet-AS (ASVAET)	**81.77** (0.20)	**72.57** (0.32)	**77.57** (0.31)	**72.31** (0.69)

Table 3: Experimental results (%). For each classifier, we performed five experiments, i.e., the supervised classifier, the supervised classifier with pre-trained embedding using unlabeled data and our model with the classifier. The results are obtained after 5 runs, and we report the mean and the standard deviation of the test accuracy, and the Macro-averaged F1 score. Better results are in bold. ♮ denotes that the results are extracted from the original paper.

put of the encoder and the decoder [5] and the out-of-vocabulary words are excluded. And it is fixed during the training. The γ is set to be 10 across the experiments.

We implemented and verified four kinds of mainstream ATSA classifiers integrated into our model, i.e., TC-LSTM (Tang et al., 2016a), MemNet (Tang et al., 2016b), BILSTM-ATT-G (Zhang and Liu, 2017), IAN (Ma et al., 2017) and TNet (Li et al., 2018b).

- **TC-LSTM**: Two LSTMs are used to model the left and right context of the target separately, then the concatenation of two representations is used to predict the label.

- **MemNet**: It uses the attention mechanism over the word embedding over multiple rounds to aggregate the information in the sentence, the vector of the final round is used for the prediction.

- **IAN**: IAN adopts two LSTMs to derive the representations of the context and the target phrase interactively and the concatenation is fed to the softmax layer.

- **BILSTM-ATT-G**: It models left and right contexts using two attention-based LSTMs and makes use of a special gate layer to combine these two representations. The resulting vector is used for the prediction.

- **TNet-AS**: Without using an attention module, TNet adopts a convolutional layer to get salient features from the transformed word representations originated from a bi-directional LSTM layer. Among current supervised models, TNet is currently one of the in-domain state-of-the-art methods and the TNet-AS is one of the two variants of TNet.

The configuration of hyper-parameters and the training settings are the same as in the original papers. Various classifiers are tested here to demonstrate the robustness of our method and show that the performance can be consistently improved for different classifiers.

4.3 Main Results

Table 3 shows the experimental results on the REST and LAPTOP datasets. Two evaluation metrics are used here, i.e., classification accuracy and Macro-averaged F1 score. The latter is more sensitive when the dataset is class-imbalance. In this table, the semi-supervised results are obtained with 10K unlabeled data. We didn't observe further improvement with more unlabeled data. The mean and the standard deviation are reported over

[5] http://nlp.stanford.edu/data/glove.8B.300d.zip

5 runs. For each classifier *clf*, we conducted the following experiments:

- *clf*: The classifier is trained using labeled data only.
- *clf* (EMB): We use CBOW (Mikolov et al., 2013) to train the word embedding vectors using both labeled and unlabeled data. And the resulting vectors, instead of pre-trained GloVe vectors, are used to initialize the embedding matrix of the classifier. This is the embedding-level semi-supervised learning as the embedding layer is trained using in-domain data.
- *clf* (ST): The self-training (ST) method is a typical semi-supervised learning method. We performed the self-training method over each classifier. At each epoch, we select the 1K samples with the best confidence and give them pseudo labels using the prediction. Then the classifier is re-trained with the new labeled data. The procedure loops until all the unlabeled samples are labeled.
- *clf* (ASVAET): The proposed method that uses *clf* as the classifier. Note again that the classifier is an independent module in the proposed model, and the same configuration is used as in the supervised learning.

Besides, we also include the results of several supervised models in the first block, i.e., CNN-ASP (Lam et al., 2018), AE-LSTM, ATAE-LSTM (Wang et al., 2016), GCAE (Li and Xue, 2018), from the original paper.

From the Table 3, the ASVAET is able to improve supervised performance consistently for all classifiers. For the MemNet, the test accuracy can be improved by about 2% by the TSSVAE, and so as the Macro-averaged F1. The TNet-AS outperforms the other three models.

Compared with the other two semi-supervised methods, the ASVAET also shows better results. The ASVAET outperforms the compared semi-supervised methods evidently. The adoption of in-domain pre-trained word vectors is beneficial for the performance compared with the Glove vectors.

4.4 Ablation Studies

4.4.1 Effect of Labeled Data

Here we investigated whether the ASVAET works with less labeled data. Without loss of generality, the MemNet is used as the basic classifier. We sampled different amount of labeled data to verify the improvement by using ASVAET. The test accuracy curve w.r.t. the amount of labeled data used is shown in Fig. 3. With fewer labeled samples, the test accuracy decreases, however, the improvement becomes more evident. When using 500 labeled samples, the improvement is about 3.2%. With full 3591 labeled samples, 1.5% gain can be obtained. This illustrates that our method can improve the accuracy with limited data.

Accuracy	w/o sharing	w/ sharing
TC-LSTM (ASVAET)	**78.34**	77.65
MemNet (ASVAET)	**80.58**	78.82
IAN (ASVAET)	**80.23**	79.22
BILSTM-ATT-G (ASVAET)	**81.11**	78.36
TNet-AS (ASVAET)	**81.77**	79.53

Table 4: Comparison between with or without sharing embedding on the REST dataset.

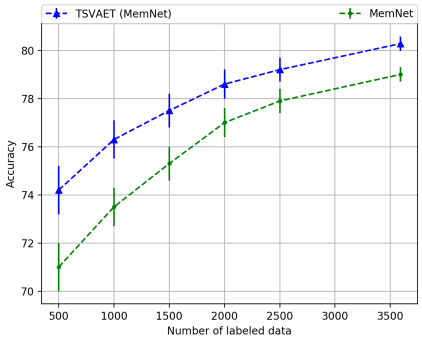


Figure 3: The test accuracy w.r.t. the number of labeled samples on the REST dataset with MemNet classifier.

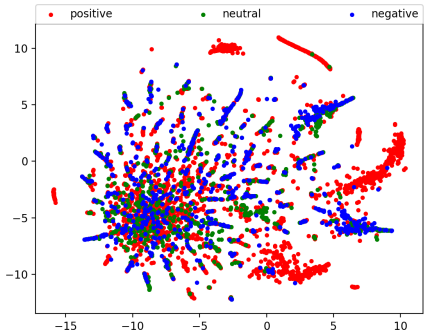


Figure 4: The distribution of the REST dataset in latent space $\mathbf{z}$ using t-SNE.

4.4.2 Effect of Sharing Embeddings

In previous works, the word embedding is shared among all the components. In other words, the word embedding is also tuned in learning to reconstruct the data. It is questionable whether the improvement is obtained by using VAE or multi-

Positive
... the best *food* i 've ever had !!! ...
... the *lox* is very tasty ...
... the *rice* is a great value ...
Negative
... the worst *food* i 've ever had !!! ...
... the *lox* is a bit of boring ...
... the *rice* is awful ...
Neutral
... had the *food* in the restaurant ...
... lox with a glass of chilli sauce ...
... the *rice* with a couple of olives salad ...

Table 5: Nice sentences that are generated by controlling the sentiment polarity y using the decoder.

task learning (text generation and classification). In the aforementioned experiments, the embedding layer is not shared between the classifier and autoencoder. This implementation guarantees that the improvement does not come from learning to generate. To verify if sharing embedding will benefit, we also conducted experiments with sharing embedding, as illustrated in Table. 4. The results indicate that the joint training for the embedding layer is negative for improving the performance in this task. The gradient from the autoencoder may collide with the gradients from the classifier and therefore, interferes with the optimization direction.

4.5 Analysis of the Latent Space

Transformer encodes the data into two representations, i.e., y and $\mathbf{z}$. These two latent variable represented sentiment polarity and lexical context individually from the input text. We expect the y and $\mathbf{z}$ are fully disentangled and represent different meanings. The scatters of latent variable $\mathbf{z}$ (cf. Fig. 4) helps us have a better understanding. As shown in the figure, the distributions of three different polarities are very similar, which indicates that the lexical context reprensetation $\mathbf{z}$ is independent of the polarity y.

The generation ability of the decoder is also investigated. Several sentences are generated and selected in the Table 5. By controlling the sentiment polarity y with the same $\mathbf{z}$, the decoder can generate sentences with different sentiment in a similar format. This indicates that the decoder is trained successfully to perceive the y and model the relationship between the y and $\mathbf{x}$.

5 Conclusion

A VAE-based framework has been proposed for the ATSA task. In this work, the encoder and decoder are constructed from the Transformers. Both analytical and experimental work has been carried out to show the effectiveness of the ASVAET. The method is verified with various kinds of classifiers. For all tested classifiers, the improvement is obtained when equipped with ASVAET, which demonstrates its universality.

In this paper, the aspect-term is assumed to be known and there is an error accumulation problem when using the pre-trained aspect extractor. According to this, in future work, it is also interesting to show if it is possible to learn the aspect and sentiment polarity jointly for the unlabeled data. It will be of great importance if detailed knowledge can be extracted from the unlabeled data, which will shed light on other related tasks.

References

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. 2019. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*.

Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. 2018. Variational sequential labelers for semi-supervised learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 215–226.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)*, Banff, Canada.

Wai Lam, Xin Li, Bei Shi, and Lidong Bing. 2018. Transformation networks for target-oriented sentiment classification. pages 946–956.

Lishuang Li, Yang Liu, and AnQiao Zhou. 2018a. Hierarchical attention based position-aware network for aspect-level sentiment analysis. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 181–189.

Tao Li and Wei Xue. 2018. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2514–2523.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018b. Transformation networks for target-oriented sentiment classification. *arXiv preprint arXiv:1805.01086*.

Fei Liu, Trevor Cohn, and Timothy Baldwin. 2018. Recurrent entity networks with delayed memory update for targeted aspect-based sentiment analysis. pages 278–283.

Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4068–4074.

Diego Marcheggiani and Ivan Titov. 2016. Discrete-state variational autoencoders for joint discovery and factorization of relations. *TACL*, 4:231–244.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Sinno Jialin Pan and Wenya Wang. 2018. Recursive neural structural correspondence network for cross-domain aspect and opinion co-extraction. pages 2171–2181.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014.*, pages 27–35.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016a. Effective lstms for target-dependent sentiment classification. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 3298–3307.

Duyu Tang, Bing Qin, and Ting Liu. 2016b. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 214–224.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 606–615.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation. *arXiv preprint arXiv:1904.12848*.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational autoencoder for semi-supervised text classification. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3358–3364.

Wei Xue and Tao Li. 2018. Aspect based sentiment analysis with gated convolutional networks. *arXiv preprint arXiv:1805.07043*.

Yue Zhang and Jiangming Liu. 2017. Attention modeling for targeted sentiment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers*, pages 572–577.

Learning to Detect Opinion Snippet for Aspect-Based Sentiment Analysis

Mengting Hu[1]* **Shiwan Zhao**[2]† **Honglei Guo**[2] **Renhong Cheng**[1] **Zhong Su**[2]
[1] Nankai University [2] IBM Research - China
mthu@mail.nankai.edu.cn, {zhaosw, guohl}@cn.ibm.com
chengrh@nankai.edu.cn, suzhong@cn.ibm.com

Abstract

Aspect-based sentiment analysis (ABSA) is to predict the sentiment polarity towards a particular aspect in a sentence. Recently, this task has been widely addressed by the neural attention mechanism, which computes attention weights to softly select words for generating aspect-specific sentence representations. The attention is expected to concentrate on opinion words for accurate sentiment prediction. However, attention is prone to be distracted by noisy or misleading words, or opinion words from other aspects. In this paper, we propose an alternative hard-selection approach, which determines the start and end positions of the opinion snippet, and selects the words between these two positions for sentiment prediction. Specifically, we learn deep associations between the sentence and aspect, and the long-term dependencies within the sentence by leveraging the pre-trained BERT model. We further detect the opinion snippet by self-critical reinforcement learning. Especially, experimental results demonstrate the effectiveness of our method and prove that our hard-selection approach outperforms soft-selection approaches when handling multi-aspect sentences.

1 Introduction

Aspect-based sentiment analysis (Pang and Lee, 2008; Liu, 2012) is a fine-grained sentiment analysis task which has gained much attention from research and industries. It aims at predicting the sentiment polarity of a particular aspect of the text. With the rapid development of deep learning, this task has been widely addressed by attention-based neural networks (Wang et al., 2016; Ma et al., 2017; Cheng et al., 2017; Tay et al., 2018; Wang et al., 2018a). To name a few, Wang et al. (2016) learn to attend on different parts of the sentence given different aspects, then generates aspect-specific sentence representations for sentiment prediction. Tay et al. (2018) learn to attend on correct words based on associative relationships between sentence words and a given aspect. These attention-based methods have brought the ABSA task remarkable performance improvement.

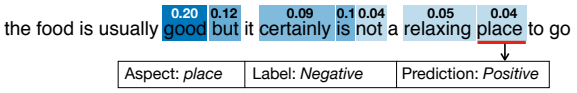


Figure 1: Example of attention visualization. The attention weights of the aspect *place* are from the model ATAE-LSTM (Wang et al., 2016), a typical attention mechanism used for soft-selection.

Previous attention-based methods can be categorized as **soft-selection** approaches since the attention weights scatter across the whole sentence and every word is taken into consideration with different weights. This usually results in attention distraction (Li et al., 2018b), i.e., attending on noisy or misleading words, or opinion words from other aspects. Take Figure 1 as an example, for the aspect *place* in the sentence *"the food is usually good but it certainly is not a relaxing place to go"*, we visualize the attention weights from the model ATAE-LSTM (Wang et al., 2016). As we can see, the words *"good"* and *"but"* are dominant in attention weights. However, *"good"* is used to describe the aspect *food* rather than *place*, *"but"* is not so related to *place* either. The true opinion snippet *"certainly is not a relaxing place"* receives low attention weights, leading to the wrong prediction towards the aspect *place*.

Therefore, we propose an alternative **hard-selection** approach by determining two positions

* Work performed while interning at IBM Research - China.
† Corresponding author.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 970–979
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

in the sentence and selecting words between these two positions as the opinion expression of a given aspect. This is also based on the observation that opinion words of a given aspect are usually distributed consecutively as a snippet (Wang and Lu, 2018). As a consecutive whole, the opinion snippet may gain enough attention weights, avoid being distracted by other noisy or misleading words, or distant opinion words from other aspects. We then predict the sentiment polarity of the given aspect based on the average of the extracted opinion snippet. The explicit selection of the opinion snippet also brings us another advantage that it can serve as justifications of our sentiment predictions, making our model more interpretable.

To accurately determine the two positions of the opinion snippet of a particular aspect, we first model the deep associations between the sentence and aspect, and the long-term dependencies within the sentence by BERT (Devlin et al., 2018), which is a pre-trained language model and achieves exciting results in many natural language tasks. Second, with the contextual representations from BERT, the two positions are sequentially determined by self-critical reinforcement learning. The reason for using reinforcement learning is that we do not have the ground-truth positions of the opinion snippet, but only the polarity of the corresponding aspect. Then the extracted opinion snippet is used for sentiment classification. The details are described in the model section.

The main contributions of our paper are as follows:

- We propose a hard-selection approach to address the ABSA task. Specifically, our method determines two positions in the sentence to detect the opinion snippet towards a particular aspect, and then uses the framed content for sentiment classification. Our approach can alleviate the attention distraction problem in previous soft-selection approaches.

- We model deep associations between the sentence and aspect, and the long-term dependencies within the sentence by BERT. We then learn to detect the opinion snippet by self-critical reinforcement learning.

- The experimental results demonstrate the effectiveness of our method and also our approach significantly outperforms soft-selection approaches on handling multi-aspect sentences.

2 Related Work

Traditional machine learning methods for aspect-based sentiment analysis focus on extracting a set of features to train sentiment classifiers (Ding et al., 2009; Boiy and Moens, 2009; Jiang et al., 2011), which usually are labor intensive. With the development of deep learning technologies, neural attention mechanism (Bahdanau et al., 2014) has been widely adopted to address this task (Tang et al., 2015; Wang et al., 2016; Tang et al., 2016; Ma et al., 2017; Chen et al., 2017; Cheng et al., 2017; Li et al., 2018a; Wang et al., 2018a; Tay et al., 2018; Hazarika et al., 2018; Majumder et al., 2018; Fan et al., 2018; Wang et al., 2018b). Wang et al. (2016) propose attention-based LSTM networks which attend on different parts of the sentence for different aspects. Ma et al. (2017) utilize the interactive attention to capture the deep associations between the sentence and the aspect. Hierarchical models (Cheng et al., 2017; Li et al., 2018a; Wang et al., 2018a) are also employed to capture multiple levels of emotional expression for more accurate prediction, as the complexity of sentence structure and semantic diversity. Tay et al. (2018) learn to attend based on associative relationships between sentence words and aspect.

All these methods use normalized attention weights to softly select words for generating aspect-specific sentence representations, while the attention weights scatter across the whole sentence and can easily result in attention distraction. Wang and Lu (2018) propose a hard-selection method to learn segmentation attention which can effectively capture the structural dependencies between the target and the sentiment expressions with a linear-chain conditional random field (CRF) layer. However, it can only address aspect-term level sentiment prediction which requires annotations for aspect terms. Compared with it, our method can handle both aspect-term level and aspect-category level sentiment prediction by detecting the opinion snippet.

3 Model

We first formulate the problem. Given a sentence $S = \{w_1, w_2, ..., w_N\}$ and an aspect $A = \{a_1, a_2, ..., a_M\}$, the ABSA task is to predict the

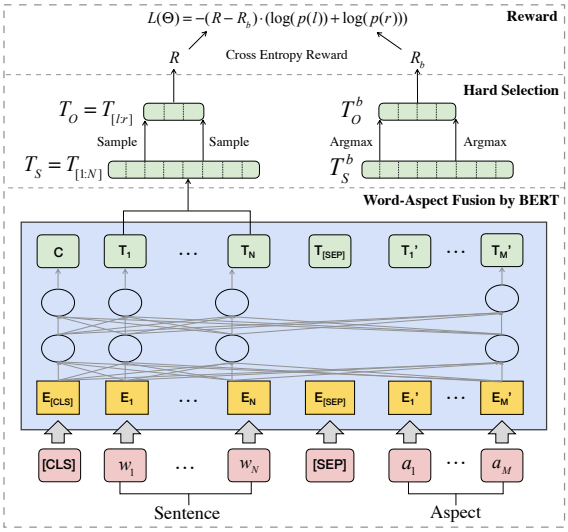


Figure 2: Network Architecture. We leverage BERT to model the relationships between sentence words and a particular aspect. The sentence and aspect are packed together into a single sequence and fed into BERT, in which E represents the input embedding, and T_i represents the contextual representation of token i. With the contextual representations from BERT, the start and end positions are sequentially sampled and then the framed content is used for sentiment prediction. Reinforcement learning is adopted for solving the non-differentiable problem of sampling.

sentiment of A. In our setting, the aspect can be either aspect terms or an aspect category. As aspect terms, A is a snippet of words in S, i.e., a sub-sequence of the sentence, while as an aspect category, A represents a semantic category with $M = 1$, containing just an abstract token.

In this paper, we propose a hard-selection approach to solve the ABSA task. Specifically, we first learn to detect the corresponding opinion snippet $O = \{w_l, w_{l+1}..., w_r\}$, where $1 \leq l \leq r \leq N$, and then use O to predict the sentiment of the given aspect. The network architecture is shown in Figure 2.

3.1 Word-Aspect Fusion

Accurately modeling the relationships between sentence words and an aspect is the key to the success of the ABSA task. Many methods have been developed to model word-aspect relationships. Wang et al. (2016) simply concatenate the aspect embedding with the input word embeddings and sentence hidden representations for computing aspect-specific attention weights. Ma et al. (2017) learn the aspect and sentence interactively by using two attention networks. Tay et al. (2018) adopt circular convolution of vectors for performing the word-aspect fusion.

In this paper, we employ BERT (Devlin et al., 2018) to model the deep associations between the sentence words and the aspect. BERT is a powerful pre-trained model which has achieved remarkable results in many NLP tasks. The architecture of BERT is a multi-layer bidirectional Transformer Encoder (Vaswani et al., 2017), which uses the self-attention mechanism to capture complex interaction and dependency between terms within a sequence. To leverage BERT to model the relationships between the sentence and the aspect, we pack the sentence and aspect together into a single sequence and then feed it into BERT, as shown in Figure 2. With this sentence-aspect concatenation, both the word-aspect associations and word-word dependencies are modeled interactively and simultaneously. With the contextual token representations $T_S = T_{[1:N]} \in \mathbb{R}^{N \times H}$ of the sentence, where N is the sentence length and H is the hidden size, we can then determine the start and end positions of the opinion snippet in the sentence.

3.2 Soft-Selection Approach

To fairly compare the performance of soft-selection approaches with hard-selection approaches, we use the same word-aspect fusion results T_S from BERT. We implement the attention mechanism by adopting the approach similar to the work (Lin et al., 2017).

$$\begin{aligned} \boldsymbol{\alpha} &= softmax(\boldsymbol{v_1} tanh(W_1 {T_S}^{\mathrm{T}})) \\ \boldsymbol{g} &= \boldsymbol{\alpha} T_S \end{aligned} \tag{1}$$

where $\boldsymbol{v_1} \in \mathbb{R}^H$ and $W_1 \in \mathbb{R}^{H \times H}$ are the parameters. The normalized attention weights $\boldsymbol{\alpha}$ are used to softly select words from the whole sentence and generate the final aspect-specific sentence representation $\boldsymbol{g}$. Then we make sentiment prediction as follows:

$$\hat{\boldsymbol{y}} = softmax(W_2 \boldsymbol{g} + \boldsymbol{b}) \tag{2}$$

where $W_2 \in \mathbb{R}^{C \times H}$ and $\boldsymbol{b} \in \mathbb{R}^C$ are the weight matrix and bias vector respectively. $\hat{\boldsymbol{y}}$ is the probability distribution on C polarities. The polarity with highest probability is selected as the prediction.

3.3 Hard-Selection Approach

Our proposed hard-selection approach determines the start and end positions of the opinion snippet

and selects the words between these two positions for sentiment prediction. Since we do not have the ground-truth opinion snippet, but only the polarity of the corresponding aspect, we adopt reinforcement learning (Williams, 1992) to train our model. To make sure that the end position comes after the start position, we determine the start and end sequentially as a sequence training problem (Rennie et al., 2017). The parameters of the network, Θ, define a policy p_θ and output an action that is the prediction of the position. For simplicity, we only generate two actions for determining the start and end positions respectively. After determining the start position, the "state" is updated and then the end is conditioned on the start.

Specifically, we define a start vector $\boldsymbol{s} \in \mathbb{R}^H$ and an end vector $\boldsymbol{e} \in \mathbb{R}^H$. Similar to the prior work (Devlin et al., 2018), the probability of a word being the start of the opinion snippet is computed as a dot product between its contextual token representation and $\boldsymbol{s}$ followed by a softmax over all of the words of the sentence.

$$\boldsymbol{\beta_l} = softmax(T_S \boldsymbol{s}) \tag{3}$$

We then sample the start position l based on the multinomial distribution $\boldsymbol{\beta_l}$. To guarantee the end comes after the start, the end is sampled only in the right part of the sentence after the start. Therefore, the state is updated by slicing operation ${T_S}^r = T_S[l:]$. Same as the start position, the end position r is also sampled based on the distribution $\boldsymbol{\beta_r}$:

$$\boldsymbol{\beta_r} = softmax(T_S^r \boldsymbol{e}) \tag{4}$$

Then we have the opinion snippet $T_O = T_S[l:r]$ to predict the sentiment polarity of the given aspect in the sentence. The probabilities of the start position at l and the end position at r are $p(l) = \boldsymbol{\beta_l}[l]$ and $p(r) = \boldsymbol{\beta_r}[r]$ respectively.

3.3.1 Reward

After we get the opinion snippet T_O by the sampling of the start and end positions, we compute the final representation $\boldsymbol{g_o}$ by the average of the opinion snippet, $\boldsymbol{g_o} = avg(T_O)$. Then, equation 2 with different weights is applied for computing the sentiment prediction $\hat{\boldsymbol{y_o}}$. The cross-entropy loss function is employed for computing the reward.

$$R = -\sum_c y^c \log \hat{\boldsymbol{y_o}}^{\boldsymbol{c}} \tag{5}$$

where c is the index of the polarity class and y is the ground truth.

3.3.2 Self-Critical Training

In this paper, we use reinforcement learning to learn the start and end positions. The goal of training is to minimize the negative expected reward as shown below.

$$L(\Theta) = -R \cdot p(l) \cdot p(r) \tag{6}$$

where Θ is all the parameters in our architecture, which includes the base method BERT, the position selection parameters $\{\boldsymbol{s}, \boldsymbol{e}\}$, and the parameters for sentiment prediction and then for reward calculation. Therefore, the *state* in our method is the combination of the sentence and the aspect. For each state, the *action* space is every position of the sentence.

To reduce the variance of the gradient estimation, the reward is associated with a reference reward or baseline R_b (Rennie et al., 2017). With the likelihood ratio trick, the objective function can be transformed as.

$$L(\Theta) = -(R - R_b) \cdot (log(p(l)) + log(p(r))) \tag{7}$$

The baseline R_b is computed based on the snippet determined by the baseline policy, which selects the start and end positions greedily by the $argmax$ operation on the $softmax$ results. As shown in Figure 2, the reward R is calculated by sampling the snippet, while the baseline R_b is computed by greedily selecting the snippet. Note that in the test stage, the snippet is determined by $argmax$ for inference.

4 Experiments

In this section, we compare our hard-selection model with various baselines. To assess the ability of alleviating the attention distraction, we further conduct experiments on a simulated multi-aspect dataset in which each sentence contains multiple aspects.

4.1 Datasets

We use the same datasets as the work by Tay et al. (2018), which are already processed to token lists and released in Github[1]. The datasets are from SemEval 2014 task 4 (Pontiki et al., 2014), and SemEval 2015 task 12 (Pontiki et al., 2015), respectively. For aspect term level sentiment classification task (denoted by T), we apply the Laptops and

[1] https://github.com/vanzytay/ABSA_DevSplits

Task	Dataset	*All*	*P*	*N*	*Nu*
T	Laptops Train	1813	767	673	373
T	Laptops Dev	500	220	193	87
T	Laptops Test	638	341	128	169
T	Restaurants Train	3102	685	1886	531
T	Restaurants Dev	500	278	120	102
T	Restaurants Test	1120	728	196	196
C	Restaurants Train	3018	1873	712	433
C	Restaurants Dev	500	306	127	67
C	Restaurants Test	973	657	222	94
C	SE 14+15 Train	3587	1069	2310	208
C	SE 14+15 Dev	427	274	134	19
C	SE 14+15 Test	1011	455	496	60

Table 1: Dataset statistics. T and C denote the aspect-term and aspect-category tasks, respectively. *P*, *N*, and *Nu* represent the numbers of instances with positive, negative and neutral polarities, and *All* is the total number of instances.

Restaurants datasets from SemEval 2014. For aspect category level sentiment prediction (denoted by C), we utilize the Restaurants dataset from SemEval 2014 and a composed dataset from both SemEval 2014 and SemEval 2015. The statistics of the datasets are shown in Table 1.

4.2 Implementation Details

Our proposed models are implemented in PyTorch[2]. We utilize the bert-base-uncased model, which contains 12 layers and the number of all parameters is 100M. The dimension H is 768. The BERT model is initialized from the pre-trained model, other parameters are initialized by sampling from normal distribution $\mathcal{N}(0, 0.02)$. In our experiments, the batch size is 32. The reported results are the testing scores that fine-tuning 7 epochs with learning rate 5e-5.

4.3 Compared Models

- **LSTM**: it uses the average of all hidden states as the sentence representation for sentiment prediction. In this model, aspect information is not used.

- **TD-LSTM** (Tang et al., 2015): it employs two LSTMs and both of their outputs are applied to predict the sentiment polarity.

- **AT-LSTM** (Wang et al., 2016): it utilizes the attention mechanism to produce an aspect-specific sentence representation. This method is a kind of soft-selection approach.

- **ATAE-LSTM** (Wang et al., 2016): it also uses the attention mechanism. The difference with AT-LSTM is that it concatenates the aspect embedding to each word embedding as the input to LSTM.

- **AF-LSTM(CORR)** (Tay et al., 2018): it adopts circular correlation to capture the deep fusion between sentence words and the aspect, which can learn rich, higher-order relationships between words and the aspect.

- **AF-LSTM(CONV)** (Tay et al., 2018): compared with AF-LSTM(CORR), this method applies circular convolution of vectors for performing word-aspect fusion to learn relationships between sentence words and the aspect.

- **BERT-Original**: it makes sentiment prediction by directly using the final hidden vector C from BERT with the sentence-aspect pair as input.

4.4 Our Models

- **BERT-Soft**: as described in Section 3.2, the contextual token representations from BERT are processed by self attention mechanism (Lin et al., 2017) and the attention-weighted sentence representation is utilized for sentiment classification.

- **BERT-Hard**: as described in Section 3.3, it takes the same input as BERT-Soft. It is called a hard-selection approach since it employs reinforcement learning techniques to explicitly select the opinion snippet corresponding to a particular aspect for sentiment prediction.

4.5 Experimental Results

In this section, we evaluate the performance of our models by comparing them with various baseline models. Experimental results are illustrated in Table 2, in which *3-way* represents 3-class sentiment classification (*positive*, *negative* and *neutral*) and *Binary* denotes binary sentiment prediction (*positive* and *negative*). The best score of each column is marked in bold.

[2]https://github.com/huggingface/pytorch-pretrained-BERT

		Term-Level				Category-Level				
		Laptops		Restaurants		Restaurants		SemEval 14+15		
Model	Aspect	3-way	Binary	3-way	Binary	3-way	Binary	3-way	Binary	Avg
LSTM	No	61.75	78.25	67.94	82.03	73.38	79.97	75.96	79.92	74.90
TD-LSTM	Yes	62.38	79.31	69.73	84.41	79.97	75.96	79.92	74.90	75.63
AT-LSTM	Yes	65.83	78.25	74.37	84.74	77.90	84.87	76.16	81.28	77.93
ATAE-LSTM	Yes	60.34	74.20	70.71	84.52	77.80	83.85	74.08	78.96	75.56
AF-LSTM(CORR)	Yes	64.89	79.96	74.76	86.91	80.47	86.58	74.68	81.60	78.73
AF-LSTM(CONV)	Yes	68.81	83.58	75.44	87.78	81.29	87.26	78.44	81.49	80.51
BERT-Original	Yes	74.57	88.25	82.66	**92.31**	**88.17**	92.37	80.50	86.84	85.71
BERT-Soft	Yes	**74.92**	**90.41**	82.68	91.98	87.05	91.92	80.02	86.75	85.72
BERT-Hard	Yes	74.10	89.55	**83.91**	**92.31**	**88.17**	**93.39**	**81.09**	**87.89**	**86.30**

Table 2: Experimental results (accuracy %) on all the datasets. Models in the first part are baseline methods. The results in the first part (except BERT-Original) are obtained from the prior work (Tay et al., 2018). Avg column presents macro-averaged results across all the datasets.

Firstly, we observe that BERT-Original, BERT-Soft, and BERT-Hard outperform all soft attention baselines (in the first part of Table 2), which demonstrates the effectiveness of fine-tuning the pre-trained model on the aspect-based sentiment classification task. Particularly, BERT-Original outperforms AF-LSTM(CONV) by 2.63%∼9.57%, BERT-Soft outperforms AF-LSTM(CONV) by 2.01%∼9.60% and BERT-Hard improves AF-LSTM(CONV) by 3.38%∼11.23% in terms of accuracy. Considering the average score across eight settings, BERT-Original outperforms AF-LSTM(CONV) by 6.46%, BERT-Soft outperforms AF-LSTM(CONV) by 6.47% and BERT-Hard outperforms AF-LSTM(CONV) by 7.19% respectively.

Secondly, we compare the performance of three BERT-related methods. The performance of BERT-Original and BERT-Soft are similar by comparing their average scores. The reason may be that the original BERT has already modeled the deep relationships between the sentence and the aspect. BERT-Original can be thought of as a kind of soft-selection approach as BERT-Soft. We also observe that the snippet selection by reinforcement learning improves the performance over soft-selection approaches in almost all settings. However, the improvement of BERT-Hard over BERT-Soft is marginal. The average score of BERT-Hard is better than BERT-Soft by 0.68%. The improvement percentages are between 0.36% and 1.49%, while on the Laptop dataset, the performance of BERT-Hard is slightly weaker than BERT-Soft. The main reason is that the datasets only contain a small portion of multi-aspect sentences with different polarities. The distraction of attention will not impact the sentiment prediction much in single-aspect sentences or multi-aspect sentences with the same polarities.

4.6 Experimental Results on Multi-Aspect Sentences

On the one hand, the attention distraction issue becomes worse in multi-aspect sentences. In addition to noisy and misleading words, the attention is also prone to be distracted by opinion words from other aspects of the sentence. On the other hand, the attention distraction impacts the performance of sentiment prediction more in multi-aspect sentences than in single-aspect sentences. Hence, we evaluate the performance of our models on a test dataset with only multi-aspect sentences.

A multi-aspect sentence can be categorized by two dimensions: the *Number* of aspects and the *Polarity* dimension which indicates whether the sentiment polarities of all aspects are the same or not. In the dimension of *Number*, we categorize the multi-aspect sentences as *2-3* and *More*. *2-3* refers to the sentences with two or three aspects while *More* refers to the sentences with more than three aspects. The statistics in the original dataset shows that there are much more sentences with *2-3* aspects than those with *More* aspects. In the dimension *Polarity*, the multi-aspect sentences can be categorized into *Same* and *Diff*. *Same* indicates that all aspects in the sentence have the same sentiment polarity. *Diff* indicates that the aspects have

Type	*Same*			*Diff*			Total
	2-3	*More*	Total	*2-3*	*More*	Total	
Number	1665	352	2017	655	327	982	2999

Table 3: Distribution of the multi-aspect test set. Around 67% of the multi-aspect sentences belong to the *Same* category.

Constructed Multi-Aspect Training Set								Total
Single		*P*		*N*		*Nu*		891
		297		297		297		
		Same			*Diff*			
Multi	2-asp	*2P*	*2N*	*2Nu*	*PN*	*PNu*	*NNu*	3600
		300	300	300	300	300	300	
	3-asp	*3P*	*3N*	*3Nu*	*2P1N*	*1P2N*	*PNNu*	
		300	300	300	300	300	300	

Table 4: Distribution of the multi-aspect training set. 2-asp and 3-asp indicate that the sentence contains two or three aspects respectively. Each multi-aspect sentence is categorized as *Same* or *Diff*.

different polarities.

Multi-aspect test set. To evaluate the performance of our models on multi-aspect sentences, we construct a new multi-aspect test set by selecting all multi-aspect sentences from the original training, development, and test sets of the Restaurants term-level task. The details are shown in Table 3.

Multi-aspect training set. Since we use all multi-aspect sentences for testing, we need to generate some "virtual" multi-aspect sentences for training. The simulated multi-aspect training set includes the original single-aspect sentences and the newly constructed multi-aspect sentences, which are generated by concatenating multiple single-aspect sentences with different aspects. We keep the balance of each subtype in the new training set (see Table 4). The number of *Neutral* sentences is the least among three sentiment polarities in all single-aspect sentences. We randomly select the same number of *Positive* and *Negative* sentences. Then we construct multi-aspect sentences by combining single-aspect sentences in different combinations of polarities. The naming for different combinations is simple. For example, *2P-1N* indicates that the sentence has two positive aspects and one negative aspect, and *P-N-Nu* means that the three aspects in the sentence are positive, negative, and neutral respectively. For simplicity, we only construct 2-asp and 3-asp sentences which are also the majority in the original dataset.

Type	*Same*	*Diff*			Total
		2-3	*More*	Total	
BERT-Original	73.33	57.10	60.86	58.35	68.42
BERT-Soft	75.31	57.25	57.19	57.23	69.39
BERT-Hard	**76.90**	**60.15**	**64.53**	**61.61**	**71.89**

Table 5: Experimental results (accuracy %) on multi-aspect sentences. The performance of the 3-way classification on the multi-aspect test set is reported.

Results and Discussions. The results on different types of multi-aspect sentences are shown in Table 5. The performance of BERT-Hard is better than BERT-Original and BERT-Soft over all types of multi-aspect sentences. BERT-Hard outperforms BERT-Soft by 2.11% when the aspects have the same sentiment polarities. For multi-aspect sentences with different polarities, the improvements are more significant. BERT-Hard outperforms BERT-Soft by 7.65% in total of *Diff*. The improvements are 5.07% and 12.83% for the types *2-3* and *More* respectively, which demonstrates the ability of our model on handling sentences with *More* aspects. Particularly, BERT-Soft has the poorest performance on the subset *Diff* among the three methods, which proves that soft attention is more likely to cause attention distraction.

Intuitively, when multiple aspects in the sentence have the same sentiment polarities, even the attention is distracted to other opinion words of other aspects, it can still predict correctly to some extent. In such sentences, the impact of the attention distraction is not obvious and difficult to detect. However, when the aspects have different sentiment polarities, the attention distraction will lead to catastrophic error prediction, which will obviously decrease the classification accuracy. As shown in Table 5, the accuracy of *Diff* is much worse than *Same* for all three methods. It means that the type of *Diff* is difficult to handle. Even though, the significant improvement proves that our hard-selection method can alleviate the attention distraction to a certain extent. For soft-selection methods, the attention distraction is inevitable due to their way in calculating the attention weights for every single word. The noisy or irrelevant words could seize more attention weights than the ground truth opinion words. Our method considers the opinion snippet as a

Aspect	Label	Method	Multi-Aspect Sentence	Prediction
appetizers	*Neutral*	BERT-Soft	the appetizers are ok, but the service is slow (0.24, 0.18, 0.06)	*Negative* ✗
		BERT-Hard	the appetizers are ok, but the service is slow (1)	*Neutral* ✓
service	*Negative*	BERT-Soft	the appetizers are ok, but the service is slow (0.12, 0.12, 0.12, 0.12, 0.15)	*Negative* ✓
		BERT-Hard	the appetizers are ok, but the service is slow (1)	*Negative* ✓

Figure 3: Visualization. The attention weights are visualized for BERT-Soft, and the selected opinion snippets are marked for BERT-Hard. The correctness of the predicted results is also marked.

consecutive whole, which is more resistant to attention distraction.

4.7 Visualization

In this section, we visualize the attention weights for BERT-Soft and opinion snippets for BERT-Hard. As demonstrated in Figure 3, the multi-aspect sentence *"the appetizers are OK, but the service is slow"* belongs to the category *Diff*. Firstly, the attention weights of BERT-Soft scatter among the whole sentence and could attend to irrelevant words. For the aspect *service*, BERT-Soft attends to the word *"ok"* with relatively high score though it does not describe the aspect *service*. This problem also exists for the aspect *appetizers*. Furthermore, the attention distraction could cause error prediction. For the aspect *appetizers*, *"but"* and *"slow"* gain high attention scores and cause the wrong sentiment prediction *Negative*.

Secondly, our proposed method BERT-Hard can detect the opinion snippet for a given aspect. As illustrated in Figure 3, the opinion snippets are selected by BERT-Hard accurately. In the sentence *"the appetizers are ok, but the service is slow"*, BERT-Hard can exactly locate the opinion snippets *"ok"* and *"slow"* for the aspect *appetizers* and *service* respectively.

At last, we enumerate some opinion snippets detected by BERT-Hard in Table 6. Our method can precisely detect snippets even for latent opinion expression and alleviate the influence of noisy words. For instance, *"cannot be beat for the quality"* is hard to predict using soft attention because the sentiment polarity is transformed by the negative word *"cannot"*. Our method can select the whole snippet without bias to any word and in this way the attention distraction can be alleviated. We also list some inaccurate snippets in Table 7. Some meaningless words around the true snippet are included, such as *"are"*, *"and"* and *"at"*. These words do not affect the final prediction. A possible explanation to these inaccurate words is that the true snippets are unlabeled and our method predicts them only by the supervisory signal from sentiment labels.

Positive Snippets	Negative Snippets
very good prompt attentive	not great bland
beautifully presented	can not eat this well
extremely tasty	unbearable conversation
as interesting as possible	no idea how to use
cool and soothing	would never go there
impressed by	not above ordinary
cannot be beat for the quality	not good

Table 6: Examples of accurate opinion snippets detected by BERT-Hard.

Inaccurate Snippets	
are very large and	and even greater food
are not terrible	tasty treat at
everyone who works	the money and said

Table 7: Examples of inaccurate opinion snippets detected by BERT-Hard.

5 Conclusion

In this paper, we propose a **hard-selection** approach for aspect-based sentiment analysis, which determines the start and end positions of the opinion snippet for a given input aspect. The deep associations between the sentence and aspect, and the long-term dependencies within the sentence are taken into consideration by leveraging the pre-trained BERT model. With the hard selection of the opinion snippet, our approach can alleviate the attention distraction problem of traditional attention-based soft-selection methods. Experimental results demonstrate the effectiveness of our method. Especially, our hard-selection approach outperforms soft-selection approaches significantly when handling multi-aspect sentences with different sentiment polarities.

6 Acknowledgement

This work is supported by National Science and Technology Major Project, China (Grant No. 2018YFB0204304).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computer Science*.

Erik Boiy and Marie-Francine Moens. 2009. A machine learning approach to sentiment analysis in multilingual web texts. *Information retrieval*, 12(5):526–558.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. 2017. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 conference on empirical methods in natural language processing (EMNLP)*, pages 452–461.

Jiajun Cheng, Shenglin Zhao, Jiani Zhang, Irwin King, Xin Zhang, and Hui Wang. 2017. Aspect-level sentiment classification with heat (hierarchical attention) network. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM)*, pages 97–106. ACM.

Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Xiaowen Ding, Bing Liu, and Lei Zhang. 2009. Entity discovery and assignment for opinion mining applications. In *Proceedings of the 15th ACM international conference on Knowledge discovery and data mining (SIGKDD)*, pages 1125–1134.

Feifan Fan, Yansong Feng, and Dongyan Zhao. 2018. Multi-grained attention network for aspect-level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3433–3442.

Devamanyu Hazarika, Soujanya Poria, Prateek Vij, Gangeshwar Krishnamurthy, Erik Cambria, and Roger Zimmermann. 2018. Modeling inter-aspect dependencies for aspect-based sentiment analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 266–270.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 151–160.

Lishuang Li, Yang Liu, and AnQiao Zhou. 2018a. Hierarchical attention based position-aware network for aspect-level sentiment analysis. In *Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL)*, pages 181–189.

Xin Li, Lidong Bing, Wai Lam, and Bei Shi. 2018b. Transformation networks for target-oriented sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira Dos Santos, Yu Mo, Xiang Bing, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *The 5th International Conference on Learning Representations (ICLR)*.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Dehong Ma, Sujian Li, Xiaodong Zhang, Houfeng Wang, Dehong Ma, Sujian Li, Xiaodong Zhang, and Houfeng Wang. 2017. Interactive attention networks for aspect-level sentiment classification. In *Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4068–4074.

Navonil Majumder, Soujanya Poria, Alexander Gelbukh, Md Shad Akhtar, Erik Cambria, and Asif Ekbal. 2018. Iarm: Inter-aspect relation modeling with memory networks in aspect-based sentiment analysis. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3402–3411.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. SemEval-2015 task 12: Aspect based sentiment analysis. In *SemEval 2015*.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. *(SemEval 2014)*, pages 27–35.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7008–7024.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Target-dependent sentiment classification with long short term memory. *CoRR, abs/1512.01100*.

Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. *Proceedings of the 2016 conference on empirical methods in natural language processing (EMNLP)*.

Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2018. Learning to attend via word-aspect associative fusion for aspect-based sentiment analysis. In *The Thirty-Second Conference on the Association for the Advance of Artificial Intelligence (AAAI)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems (NIPS)*, pages 5998–6008.

Bailin Wang and Wei Lu. 2018. Learning latent opinions for aspect-level sentiment classification. In *The Thirty-Second Conference on the Association for the Advance of Artificial Intelligence (AAAI)*.

Jingjing Wang, Jie Li, Shoushan Li, Yangyang Kang, Min Zhang, Luo Si, and Guodong Zhou. 2018a. Aspect sentiment classification with both word-level and clause-level attention networks. In *Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*.

Shuai Wang, Sahisnu Mazumder, Bing Liu, Mianwei Zhou, and Yi Chang. 2018b. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 957–967.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing (EMNLP)*, pages 606–615.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Multi-Level Sentiment Analysis of PolEmo 2.0: Extended Corpus of Multi-Domain Consumer Reviews

Jan Kocoń
Wrocław University of Science and Technology
Wrocław, Poland
jan.kocon@pwr.edu.pl

Piotr Miłkowski
Wrocław University of Science and Technology
Wrocław, Poland
piotr.milkowski@pwr.edu.pl

Monika Zaśko-Zielińska
University of Wrocław
Institute of Polish Studies
Wrocław, Poland
monika.zasko-zielinska@uwr.edu.pl

Abstract

In this article we present an extended version of PolEmo – a corpus of consumer reviews from 4 domains: medicine, hotels, products and school. Current version (PolEmo 2.0) contains 8,216 reviews having 57,466 sentences. Each text and sentence was manually annotated with sentiment in 2+1 scheme, which gives a total of 197,046 annotations. We obtained a high value of Positive Specific Agreement, which is 0.91 for texts and 0.88 for sentences. PolEmo 2.0 is publicly available under a Creative Commons copyright license. We explored recent deep learning approaches for the recognition of sentiment, such as Bidirectional Long Short-Term Memory (BiLSTM) and Bidirectional Encoder Representations from Transformers (BERT).

1 Introduction

In recent years, we have observed a growing interest in methods of effective sentiment analysis, especially in subjective, opinion-forming online texts. This trend is perfectly illustrated by Figure 1, which compares the popularity of two terms: *customer feedback* and *sentiment analysis*. A very dynamic growth has been observed since 2010, which correlates with the increase in the number of scientific research in this area. Many studies focus on the perception of emotion and sentiment in text messages and, for example, their impact on election results (Ramteke et al., 2016), prediction of future events (Zhang and Skiena, 2010) and security issues around the world (Subramaniyaswamy et al., 2017; Al-Rowaily et al., 2015). Automatic sentiment analysis systems have proven to be effective in analyzing many different types of text data such as emails, blogs, news, tweets and books (Medhat et al., 2014). The introduction of advanced computational techniques (machine learning, deep learning) in natural language processing has resulted in a significant increase in sentiment analysis techniques (Zhang et al., 2018). This increase for some languages is effectively limited by the lack of good quality resources for this task, especially in the form of manually annotated corpora (Balahur and Turchi, 2012; Dashtipour et al., 2016).

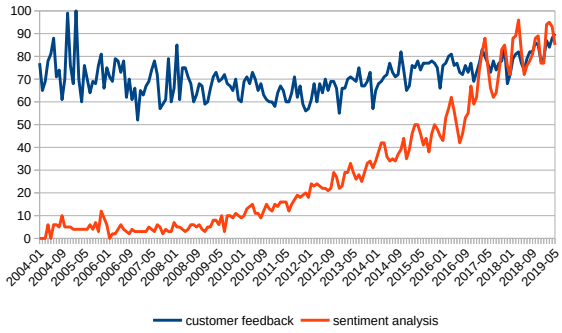


Figure 1: Google Trends (https://trends.google.com) data showing interest in time for search terms "customer feedback" and "sentiment analysis". On the vertical axis 100 means biggest search term popularity.

Analysis of the existing language resources in the area of sentiment analysis shows that they largely concern the English language (Dashtipour et al., 2016). However, there is a clear growing interest in other languages, often much more complex than English (e.g. Slavic languages in the area of loose syntax and rich inflection) and new resources become available for them, e.g., Slovene (Bučar et al., 2018), Czech (Habernal and Brychcín, 2013) or Russian (Rogers et al., 2018). Due to a small number of available corpora manually annotated with sentiment for the Polish language, we decided that the construction of the PolEmo resource will be a valuable contribution to the collection of publicly available resources for sentiment analysis and may in the future provide a basis for the creation of shared tasks, in which the recognition of sentiment for the Polish language

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 980–991
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

will also be included. Both for the construction of the corpus and for further research, we used the experience from the work on the manual annotation of the Polish WordNet – plWordNet 4.0 Emo (Janz et al., 2017; Kocoń et al., 2018a,b) – as a result of which the sentiment metadata of more than 55,000 lexical units were described.

The main objectives of the article are to present:

- The current state of resources related to the analysis of sentiment for the Polish language;
- The method of selecting data for the PolEmo 2.0 corpus, the annotation method, the annotation results and the analysis of annotation errors;
- The results of research related to the automatic analysis of sentiment, with particular emphasis on the importance of the *text domain* in this topic.

The key contribution of these studies includes:

- Detailed description of the procedure of building PolEmo 2.0: manually annotated corpus of consumer reviews from 4 domains (medicine, school, hotels, products) at 2 levels of sentiment granularity (document, sentence);
- Detailed analysis of manual annotation with regard to frequently occurring errors;
- Development of methods based on deep learning (BiLSTM, BERT), adapted to PolEmo 2.0 corpus, also using sentiment lexicon generated from plWordNet 4.0 Emo;
- Performing tests on sets prepared for the analysis of the quality of methods (1) evaluated on texts within a given domain, (2) evaluated on texts from various domains (3) trained on texts that do not include a given domain and tested on a given domain;
- Comparison of deep learning methods with classic methods (Logistic Regression), especially in the context of the ability to generalize the problem of recognizing sentiment and providing semantic representation, which is as independent of the domain as possible;
- Making PolEmo 2.0 corpus available under an open license.

2 Related Work

There are several well-known resources annotated with sentiment for English, e.g.: MPQA 3.0 (Deng and Wiebe, 2015), the Stanford Sentiment Treebank (Socher et al., 2013), Amazon Product Data (He and McAuley, 2016), Pros And Cons Dataset (Ganapathibhotla and Liu, 2008), corpora developed within the Semantic Evaluation workshops (Nakov et al., 2016; Pontiki et al., 2016), SentiWordNet (Baccianella et al., 2010) or Opinion Lexicon (Hu and Liu, 2004). There are also different approches and tools used for multilingual sentiment analysis (Lo et al., 2017) which are based on transformations on the existing resources. In this section we are focusing on the resources prepared directly for Polish.

2.1 Polish Sentiment Corpora

There are corpora for the Polish language that can be used for automatic sentiment analysis. One of them is a corpus prepared for the sentiment recognition shared task within *PolEval2017*[1] workshop (Wawer and Ogrodniczuk, 2017). The corpus contains 1550 sentences annotated at the level of phrases determined by the dependency parser. The sentences came from consumer reviews and covered 3 domains: *perfume*, *clothing* and *other*. Each node of the dependency tree received one of the three sentiment annotations: -1 (negative), 0 (neutral), 1 (positive). Most of the systems participating in the PolEval2017 competition used Tree LSTM adapted to dependency trees, including the best system, which reached an accuracy of 79% on this data.

Another resource is *HateSpeech*[2] corpus containing 2,000 posts crawled from public Polish web. These texts were annotated for hate speech. The annotator team reached an agreement score of Krippendorff's $\alpha = 0.6$ (Krippendorff, 2018). The SVM model trained on a subset of 1500 texts (containing equal amounts of hate speech and non-hate speech) obtained the precision of 0.8 (Troszyński and Wawer, 2017).

Other interesting resource is the *Polish Corpus of Suicide Notes* (PCSN) (Zaśko-Zielińska, 2013). The PCSN is one of very few such resources in the world. It includes 1,244 genuine SNs that have been scanned and manually transcribed. Each SN

[1] http://2017.poleval.pl/index.php/tasks/

[2] http://zil.ipipan.waw.pl/HateSpeech

was linguistically annotated on several levels, including selected semantic and pragmatic phenomena (Zaśko-Zielińska, 2013). The annotation is stored in a TEI-based format (Marcińczuk et al., 2011) with corrected version in a separate layer. PCSN includes also a subcorpus of 334 counterfeited SNs (elicited). They were created by volunteers who were asked to imitate a real SN for imaginary person whose characteristic had been provided at the beginning of the experiment. Most volunteers were told that the notes written by them would be used *to deceive* the computer program. Due to the sensitive nature of the texts and legal obligations of the author, the corpus is not publicly available. In the experiment described in article (Piasecki et al., 2017) we have collected 3,200 texts from the Internet as examples of non-letters. Using SVM with a rich set of features we obtained 90,06% (F1-score) in the task of distinguishing between genuine SNs, counterfeited SNs and non-letters.

2.2 Polish Sentiment Lexicons

One of the largest Polish sentiment lexical resources in terms of number of annotations is *plWordNet 4.0 Emo*[3] (Janz et al., 2017; Kocoń et al., 2018a). This dataset is available under the WordNet 3.0 license. It was built within CLARIN-PL[4] project (Piasecki, 2014). The manual annotation is done at the level of lexical units (Zaśko-Zielińska et al., 2015). Available values for polarity are: *strong negative, weak negative, neutral, weak positive, strong positive, ambiguous*. One annotator could assign only one of these values for a single lexical unit. There are more than 83,000 annotations covering more than 54,000 lexical units and 41,000 synsets (Kocoń et al., 2018b). About 22,000 of the polarity annotations are different than neutral and these annotations cover 13,000 lexical units and 9,000 synsets (22% of all synsets containing annotated units). plWordNet 4.0 Emo is used in the research presented in this article as a knowledge base for the sentiment recognition task.

Another lexicon is the *Nencki Affective Word List (NAWL)*[5] (Wierzba et al., 2015; Riegel et al., 2015). It is a database of Polish words suitable for studying various aspects of language and emotions. 2902 Polish words from the NAWL were presented to 265 subjects, who were instructed to rate them according to the intensity of each of the five basic emotions: happiness, anger, sadness, fear and disgust. The total number of ratings was 385,575.

The next resource is called the *Polish Sentiment Dictionary*[6] (Wawer, 2012; Wawer and Rogozinska, 2012). It contains 3,704 words with sentiment scores computed using supervised methods presented in (Wawer and Rogozinska, 2012).

Recently, a new resource has appeared in the Sentimenti project, containing a large database of annotated lexical units and annotated texts. Details are described in Section 2.3.

2.3 Sentimenti Project

This year, the first results of the Sentimenti[7] project (Kocoń et al., 2019a) were published, which were aimed at creating methods of analyzing texts written on the Internet in terms of emotions aroused by the recipients of the analysed content. A large database has been created, in which 30,000 lexical units from plWordNet database (Piasecki et al., 2014) and 7,000 texts were annotated. Most of the texts were consumer reviews from the domain of hotels and medicine. The elements were annotated by 20,000 unique Polish respondents in the Computer Assisted Web Interview survey and more than 50 marks were obtained for each element. Within each mark, polarisation of the element, stimulation and basic emotions aroused by the recipients are determined. The total number of manual annotations is 3,742,611 for texts and 19,141,041 for lexical units The first results concerning the automatic recognition of polarity and emotions for this set are presented in (Kocoń et al., 2019a) and propagation of this annotation with the use of Heterogeneous Structured Synset Embeddings is presented in (Kocoń et al., 2019b). Due to the commercial nature of the Sentimenti project, it is planned to publish only 20% of the project data available soon. The data will be published at the main project's site[7].

The Sentimenti project has interested both the scientific community and business. Within the CLARIN-PL project, we decided that in addition to a large annotated plWordNet lexicon, there

[3] http://plwordnet.pwr.edu.pl
[4] https://clarin-pl.eu
[5] https://exp.lobi.nencki.gov.pl/nawl-analysis
[6] http://zil.ipipan.waw.pl/SlownikWydzwieku
[7] https://sentimenti.com/

should also be a large corpus annotated with sentiment, available under an open license. In the next part we present the works related to the preparation of PolEmo.

3 PolEmo Sentiment Corpus

3.1 Motivation

Linguistic research on sentiment recognition involves two approaches: (1) *bottom-up* from the perspective of analysing the occurrence of emotional words and (2) *top-down* from the perspective of the entire document. The first attempt is usually a consequence of the creation of the sentiment lexicon, e.g. manual annotation of the WordNet (Baccianella et al., 2010). The second results from the analysis of the specific text content in which we see that the sentiment of a word or phrase changes under the influence of the surrounding context (Taboada et al., 2008). This change may vary depending on the domain of the text.

A discourse perspective in sentiment analysis is an attempt to address limitations of *bottom-up* methods (e.g. problems with negation, focusing on adjectives). It used findings of Rhetorical Structure Theory (Mann and Thompson, 1988). The attempt bears in mind local and global orientation in the text, discourse structure or topicality (Taboada et al., 2008). It allows the researcher to extract the most important sentences from the text in the perspective of the entire discourse context: nucleus satellite method (Wang et al., 2012). The relevance of the sentences is evaluated in relation to the main topic and the analysis omits some less important parts of the text.

There are interesting articles focused at domain-oriented sentiment analysis (Kanayama and Nasukawa, 2006), where a system is trained on labeled reviews from one source domain but is meant to be deployed on another (Glorot et al., 2011). The latter article describes the research carried out on the Amazon Product Data (He and McAuley, 2016). The ratings were assigned to reviews by authors of the reviews. Moreover, the ratings were applied to the entire text. Our idea was to obtain such a set of reviews that would be rated by the recipients and not by the authors of the content. The annotation should take into account not only the level of the entire review, but also the level of the individual sentences of the review. Additionally, this dataset was supposed to be a multi-domain one, to evaluate potential knowledge transfer across domains.

ID	Name	Source	Author	Subject
H	hotels	tripadvisor.com	visitor	hotel
M	medicine	znanylekarz.pl	patient	doctor
S	school	polwro.pl	student	teacher
P	products	ceneo.pl	buyer	product

Table 1: Each review is described in its domain **ID** and domain **Name** with the given **Source** of the review, **Author**'s type and the general **Subject** of the review.

3.2 Dataset

In the initial part of the work, presented in article (Kocoń et al., 2019), we have chosen online customer reviews from four domains, presented in Table 1. At the beginning of our work we had only 1000 texts for each of the following domains: *school, products, medicine*. In the case of *product* reviews, we also had metadata from the reviewer, how many stars he assigned to a specific review (from 1 to 5, where 5 means the most positive review). We used this information to select the reviews for the corpus, where 200 reviews from each star category were added.

On the basis of a preliminary analysis of several dozen examples of opinions, we have come to the conclusion that neutral examples are very difficult to find in the case of reviews. In the meantime, the corpus was extended by 8000 texts from the category Medicine and 17000 texts from the category Hotels, also with a uniform distribution in relation to the star categories available in the source data (also 1 to 5). In order to capture the phenomenon of neutral text, we decided to add 2000 new texts to each of the last two fields (medicine, hotels). These texts were fragments of articles from information portals on hotel industry[8] and health[9].

In Section 3.3 we present how the genre structure of a customer review affects the text sentiment polarity. It is an enhancement of the discourse perspective in sentiment analysis.

3.3 Pilot Annotation

Our CLARIN-PL pilot study on sentiment analysis of customer reviews was conducted in 2018. The initial part of the analysis included 3,000 reviews. Each text was manually annotated by two annotators: a psychologist and a linguist,

[8] http://ehotelarstwo.com
[9] http://naukawpolsce.pap.pl/zdrowie

who worked according to the general guidelines. The annotation tool used for this task was Inforex[10] (Marcińczuk et al., 2012; Marcińczuk and Oleksy, 2019) – a web-based system for text corpora management, annotation and analysis, available as an open source project. In the pilot project, we decided to deal with the sentiment annotation of the entire text. There was also an attempt to manually extract descriptions of particular aspects of the review. In both annotation cases we used the same tag system that is used in plWordNet Emo for lexical units: [+m] (strong positive), [+s] (weak positive), [-m] (strong negative), [-s] (weak negative), [amb] (ambiguous). We assumed that reviews are always characterised by a certain polarity, which is why we did not use the [0] (neutral) tag in the pilot annotation.

In the process of annotation we focused mainly on the strategic places of the text. In the consumer review these are opening and closing sentences, i.e. a text frame. The opening sentences consist of the general opinion of the author about the subject of the review, and the closing sentences contain the author's recommendation for the review recipients. The annotators have developed their first overall rating based on these two segments. In the text, review authors changed their opinions only subtly. Regardless of the modification of the main opinion in the text, we did not use the [amb] tag when the frame of the text was clearly positive or negative. Polarity of the text frame was influenced not only by the lexical content, but also by non-verbal elements: emoticons or multiplication of punctuation marks, e.g. exclamation marks.

The annotators were also recommended to distinguish those parts of the text that are placed in one sentence, but relate to different aspects (e.g. the teacher's appearance or teaching skills). This task turned out to be very difficult, specially in specifying, even with the help of guidelines, how to mark precisely in the text the boundaries of a given aspect. The Positive Specific Agreement (Hripcsak and Rothschild, 2005) between the annotators in the task of annotating the boundaries of aspects was below 0.15. The concept of annotation was radically changed and presented in Section 3.4.

[10] https://github.com/CLARIN-PL/Inforex

3.4 PolEmo Annotation Guidelines

In the main stage of the project we decided to annotate the sentiment for the whole text (a *meta* level) and the *sentence* level. We assumed that this strategy allows to establish the acceptable value of PSA, because the division of the text into sentences was determined by the MACA[11] tool (Radziszewski and Śniatowski, 2011), so the task was limited only to annotating the sentiment of the sentence. We followed the rule that the *meta* annotation results partially from sentence annotations, however the frame polarity is the main factor for the final meta annotation. We have prepared the following annotation tags, regardless of whether the entire text or sentence is annotated:

- SP – entirely positive;
- WP – generally positive, but there are some negative aspects within the review;
- 0 – neutral;
- WN – generally negative, but there are some positive aspects within the review;
- SN – entirely negative;
- AMB – there are both positive and negative aspects in the text that are balanced in terms of relevance.

This time we used [0] tag (neutral) because in the main stage of the project we extended the corpus with neutral texts presented in Section 3.2. Also reviews that are not neutral often contain neutral sentences.

We tested the new guidelines on a subset of 50 documents, achieving a PSA of 80% for the meta level and 78% for the sentence level. In the second iteration of the annotation guidelines improvement, the values were 87% (meta) and 85% (sentence). In the last iteration of the improvement of the guidelines, the annotators reached a PSA of 90% (meta) and 87% (sentence).

3.5 PolEmo 2.0 Annotation Analysis

We decided to publish the first results of the research on the PolEmo 1.0 corpus when the number of annotated reviews reached 8462 and the number of annotated sentences was 35724 (Kocoń et al.,

[11] Morphological Analysis Converter and Aggregator: http://nlp.pwr.edu.pl/redmine/projects/libpltagger/wiki

2019). Due to the fact that in PolEmo 2.0 there are only those annotated elements that received 2 annotations from linguists and were agreed by the super-annotator, this time we provide 8216 reviews and 57466 sentences. In Section 5 we present Table 7 with the final distribution of annotations and Table 6 with the number of elements in each domain (evaluation data splits). In this section we focus on annotation agreement and annotation errors.

L	D	SN	WN	0	WP	SP	AMB	A
T	H	91.91	36.29	99.41	39.38	91.61	40.11	79.73
	M	94.09	26.42	99.05	22.37	96.28	42.46	89.52
	P	94.06	23.33	100.0	47.62	85.95	33.68	78.76
	S	87.50	20.00	00.00	36.07	92.52	54.19	77.03
	A	92.87	32.20	99.18	37.10	93.48	41.86	83.41
S	H	93.78	00.00	88.40	00.20	93.05	33.94	85.39
	M	90.43	28.75	91.84	26.58	93.43	39.04	88.83
	P	91.27	01.20	48.42	06.90	90.84	30.50	76.82
	S	79.21	00.00	26.56	02.76	81.39	33.73	60.78
	A	91.93	11.94	87.21	07.24	92.12	33.86	84.56

Table 2: Positive Specific Agreement for annotations obtained at the level (L) of text (T) and sentence (S) for each domain (D): hotels (H), medicine (M), products (P), school (S) and all (A).

Table 2 presents PSA values obtained at the level of text and sentence for all domains. The overall PSA value for texts is 83.41% and for sentences is 84.56%. It is worth noting that for the domains to which we have not added neutral texts (products, school), there are practically no neutral annotations at the text level (see Table 7). The highest values are obtained for the most obvious categories (SP, SN and 0), regardless of the level of text description. For the remaining categories PSA value is lower than 40.00% in most cases.

D	A/ WP	SN/ WN	SP/ WP	A/ WN	A/ SN	A/ SP	R R	A/WP/ WN
H	28.55	22.07	18.33	17.08	07.86	03.12	02.99	47.63
M	18.66	26.24	14.29	17.49	12.24	04.37	06.71	37.32
P	28.16	24.27	13.59	19.42	10.68	02.91	00.97	48.54
S	36.21	07.76	28.45	10.34	06.03	08.62	02.59	49.14
A	26.69	22.07	17.82	16.79	09.02	03.89	03.74	45.23

Table 3: Distribution (%) of disagreements between annotators at the text level. A – AMB tag, A/WP – one annotator assigned [AMB], other – [WP]. R is the rest of rare occurring combinations. A/WP/WN is the sum of A/WP, A/WN and WN/WP.

Table 3 presents the distribution of disagreements between annotators at the text level. The most common disagreement is within the pair of tags [AMB/WP]. Nearly half of the disagreements are related to any pair of AMB, WP and WN tags. This suggests that annotators, despite the guidelines, have difficulty in judging the relevance of aspects regardless of the domain, or it is a very subjective task.

D	SN/ 0	A/ SN	A/ 0	A/ WP	SP/ 0	A/ SP	SP/ WP	R R	A/WP/ WN
H	10.52	14.29	05.65	19.80	09.42	07.88	09.31	04.30	30.40
M	34.66	08.10	05.02	04.98	15.93	03.32	06.68	04.35	11.62
P	07.84	21.08	33.57	06.21	05.57	09.00	05.17	02.15	09.93
S	04.63	13.90	26.59	08.66	06.45	20.44	12.19	02.01	12.49
A	16.22	13.80	13.23	11.69	10.20	08.20	08.08	18.58	19.07

Table 4: Distribution (%) of disagreements between annotators at the sentence level. A – AMB tag, A/WP – one annotator assigned [AMB], other – [WP]. R is the rest of rare occurring combinations. A/WP/WN is the sum of A/WP, A/WN and WN/WP.

Table 4 presents the distribution of disagreements between annotators at the sentence level. The most common disagreement is within the pair of tags [SN/0]. This time the cases of disagreements between A/WP/WN tags are less than 20%. Most of the errors are related to the neutral sentence marking. The analysis of specific cases and a discussion with linguists showed that in the task of annotating sentences it is difficult to isolate a sentence from the context and sometimes the annotation of the next sentence was a consequence of the sentiment of the previous sentence.

We have found that it is difficult to decide on the relevance of the aspects and without creating a hierarchy of relevance of aspects for a given domain it will be hard to achieve better agreement for WP/WN/AMB tags. Due to the fact that mistakes are often within these tags, we have combined them into one AMB tag. PolEmo 2.0 will also be available for the original tags, but research (Kocoń et al., 2019) has shown that machine learning methods achieve F-score for WP/WN/AMB classes no higher than PSA. The evaluation data in this research has WP/WN/AMB tags merged into one AMB tag. Table 5 presents PSA values after the merging step. The total PSA increased from 83% to 91% for texts and from 85% to 88% for sentences.

4 Multi-Level Sentiment Recognition

Recently deep neural networks show relatively good performance among all available methods of processing such information (Glorot et al., 2011). Possibility of retrieving data from different sources like social networks (Pak and Paroubek, 2010), publicly available discussion boards or

L	D	SN	0	AMB	SP	A
T	H	91.92	99.42	78.50	91.62	89.39
	M	94.09	99.05	70.25	96.28	93.43
	P	94.06	100.0	77.82	85.95	89.07
	S	87.50	00.00	80.78	92.52	88.32
	A	92.87	99.18	76.87	93.48	90.91
S	H	93.78	88.40	65.64	93.05	89.83
	M	90.43	91.84	59.40	93.43	90.13
	P	91.27	48.42	41.22	90.84	79.12
	S	79.21	26.56	45.48	81.39	65.68
	A	91.92	87.21	56.82	92.12	87.50

Table 5: Positive Specific Agreement for annotations with WP/WN/AMB merged into one AMB tag, obtained at the level (L) of text (T) and sentence (S) for each domain (D): hotels (H), medicine (M), products (P), school (S) and all (A).

marketing platforms connected with proper annotations on training data set can provide not only simple positive, negative or neutral classification but lead to accurate fine-grained sentiment prediction (Guzman and Maalej, 2014).

We selected the same classifiers for the recognition tasks as in (Kocoń et al., 2019): (1) Logistic Regression as a fastText recognition model (Joulin et al., 2017) with KGR10 word embeddings (Kocoń and Gawor, 2018) providing a baseline for text classification; (2) BiLSTM (Zhou et al., 2016) in two variants: KGR10 embeddings as features only and KGR10 embeddings extended with general polarity information from sentiment dictionary described in (Kocoń et al., 2019); (3) BERT (Devlin et al., 2018) with additional sequence classification layer.

We changed the architecture of BiLSTM and BERT architecture. In case of BiLSTM, instead of fixed input length we changed the model to work with text of any length. The input tensor shape is (None, 300) for *embedding-only* variant (BiLSTM) and (None, 306) for *embedding+dictionary* variant (BiLSTMd). We changed the shape of the second gaussian noise layer to (None, 300)/(None, 306), respectively. Next layers remain the same, i.e. (1) BiLSTM layer with 1024 hidden units, (2) dropout layer (ratio 0.2). Last dense layer changed due to the reduction of sentiment labels from 6 to 4 by label merging process described in Section 3.5. For BERT we used the same architecture as in (Kocoń et al., 2019) for the whole texts, but we changed it for sentences. We reduced the maximum sequence length from 512 to 64 (covers more than 99% of sentences) and we increased batch size from 32 to 128.

5 Evaluation

As in article (Kocoń et al., 2019a; Kocoń et al., 2019), we prepared three variants of evaluation of the sentiment classification methods:

- *SD – Single Domain* – evaluation sets created using elements from the same domain;
- *DO – Domain Out* – train/dev sets created using elements from 3 domains, test set from the remaining domain. This variant allows to evaluate the ability of the classification method to capture the domain-independent sentiment features;
- *MD – Mixed Domains* – SD train/dev/test sets joined respectively. This variant allows to examine the ability of the classifier to generalise the task of sentiment analysis in all available domains.

We use *SDT*, *DOT*, and *MDT* abbreviations for *text* evaluation types and *SDS*, *DOS*, and *MDS* for *sentence* evaluation types. We use also prefixes of domains (*Hotels, Medicine, School, Products*) as suffixes for *SD** and *DO** variants, e.g. SDS-H is a Single Domain evaluation type performed on Sentences within *Hotels* domain, whereas DOT-M is a Domain-Out evaluation type performed on Texts trained on texts outside *Medicine* domain and tested on texts from that domain.

Table 6 shows the number of texts and sentences annotated by linguists for all evaluation types, with division into the number of elements within training, validation and test sets. The distribution of labels for each domain (both texts and sentences) is presented in Table 7.

6 Results

Table 8 presents the values of F1-score for each label, global F1-score, micro-AUC and macro-AUC for all evaluation types related to the texts. In case of evaluation for a single domain for each label, fastText (using Logistic Regression) outperformed other classifiers in 16 out of 28 distinguishable cases. The worst results are obtained for *ambiguous* cases, but in 9 out of 13 cases F1-score is higher than 0.5 and this result is much better, than obtained for intermediate labels (*weak* positive and *weak* negative) presented in work (Kocoń

Type	Domain	Train	Dev	Test	SUM
SDT	Hotels	3165	396	395	3956
	Medicine	2618	327	327	3272
	Products	387	49	48	484
	School	403	50	51	504
DOT	!Hotels	3408	427	-	3835
	!Medicine	3955	496	-	4451
	!Products	6186	774	-	6960
	!School	6170	772	-	6942
MDT	All	6573	823	820	8216
SDS	Hotels	19881	2485	2485	24851
	Medicine	18126	2265	2266	22657
	Products	5942	743	742	7427
	School	2025	253	253	2531
DOS	!Hotels	26093	3262	-	29355
	!Medicine	27848	3481	-	31329
	!Products	40032	5004	-	45036
	!School	43949	5494	-	49443
MDS	All	45974	5745	5747	57466

Table 6: The number of texts/sentences for each evaluation type in train/dev/test sets.

Type	Domain	SP	AMB	0	SN
SDT	Hotels	25.61	24.29	10.77	39.33
	Medicine	29.37	09.57	24.11	36.95
	Products	11.16	27.48	00.41	60.95
	School	51.39	38.29	00.00	10.32
	All	27.84	19.47	14.81	37.88
SDS	Hotels	29.55	12.26	17.05	41.15
	Medicine	23.18	06.26	39.48	31.08
	Products	24.61	19.86	09.36	46.17
	School	35.56	37.38	08.89	18.17
	All	26.67	11.98	24.54	36.81

Table 7: The distribution (%) of annotations in a given domain for the following sets: SDT – single domain texts (100%=8216), SDS – single domain sentences (100%=57466).

et al., 2019). BERT classifier performs much better (14 out of 28 cases) in domain-out knowledge transfer evaluation (DOT). For this evaluation type only 4 times fastText was better. These observations are consistent with the results of article (Kocoń et al., 2019a) for *valence* dimensions.

7 Conclusions

BERT's performance is below the expectations of this advanced method in case of the classification of the whole texts. Looking at both tables (8 and 9), BERT's results are the best in 64 out of 182 label-specific cases. BiLSTM outperformed other methods in 48 cases. Adding an external sentiment dictionary helped in 40 label-specific cases. Overall BiLSTM performance is better in 88 out of 182 cases. BERT dominance (when distinguishing between BiLSTM and BiLSTMd) is observed in DOT and all sentence cases. MDT case is the most promising in terms of the further use of the recognition method in applications such as brand monitoring or early crisis detection. The values of the general F1, micro AUC and macro AUC are the highest for BiLSTM variants (see Table 6).

We published PolEmo 2.0 in CLARIN-PL DSpace repository[12] under the Creative Commons 4.0 License. We also intend to test the contextualized embedding that we are currently building using the ELMo deep word representations method (Peters et al., 2018), with the use of the large KGR10 corpus presented in work (Kocoń et al., 2019a). We also want to train the basic BERT model with the use of KGR10 to investigate whether it will improve the quality of sentiment recognition. It is also very interesting to use the propagation of sentiment annotation in WordNet (Kocoń et al., 2018a,b), to increase the coverage of the sentiment dictionary and to potentially improve the recognition quality as well. This objective can be achieved by other complex methods such as OpenAI GPT-2 (Radford et al., 2019) and domain dictionaries construction methods utilising WordNet (Kocoń and Marcińczuk, 2016).

[12] http://hdl.handle.net/11321/710

References

Khalid Al-Rowaily, Muhammad Abulaish, Nur Al-Hasan Haldar, and Majed Al-Rubaian. 2015. Bisal–a bilingual sentiment analysis lexicon to analyze dark web forums for cyber security. *Digital Investigation*, 14:53–62.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

Alexandra Balahur and Marco Turchi. 2012. Multilingual sentiment analysis using machine translation? In *Proceedings of the 3rd workshop in computational approaches to subjectivity and sentiment analysis*, pages 52–60. Association for Computational Linguistics.

Jože Bučar, Martin Žnidaršič, and Janez Povh. 2018. Annotated news corpora and a lexicon for sentiment analysis in slovene. *Language Resources and Evaluation*, 52(3):895–919.

T	C	SP	AMB	0	SN	F1	micro	macro
SDT-H	1	83.58	55.56	**98.80**	85.47	80.25	94.28	73.96
	2	**87.31**	64.24	97.56	88.44	**84.05**	95.44	75.87
	3	84.69	**67.39**	96.30	**89.97**	**84.05**	**96.71**	**76.77**
	4	83.50	59.88	93.83	86.90	81.01	95.62	74.94
SDT-M	1	**82.83**	**36.84**	**98.65**	81.48	83.18	95.62	73.35
	2	78.35	18.18	96.60	78.29	77.37	92.99	70.83
	3	75.13	15.87	94.67	76.19	74.31	91.92	70.13
	4	80.75	00.00	97.30	**85.61**	**83.79**	**96.37**	**74.29**
SDT-P	1	**40.00**	**54.55**	00.00	**85.29**	**75.00**	**93.09**	**63.65**
	2	00.00	00.00	00.00	82.93	70.83	87.49	35.50
	3	00.00	08.70	00.00	67.65	50.00	77.82	44.43
	4	00.00	00.00	00.00	84.34	72.92	89.21	39.81
SDT-S	1	**81.36**	**66.67**	00.00	**50.00**	**74.00**	**84.27**	**59.85**
	2	65.31	60.47	00.00	25.00	60.00	76.92	56.23
	3	72.73	57.89	00.00	28.57	64.00	76.12	53.97
	4	71.79	00.00	00.00	00.00	56.00	79.48	51.02
DOT-H	1	77.63	41.77	**90.48**	80.85	73.16	90.39	71.30
	2	74.37	25.00	85.71	73.28	66.08	85.96	67.75
	3	82.52	**52.69**	86.42	82.14	**76.46**	92.77	73.17
	4	**83.84**	47.27	85.71	**83.43**	76.20	**94.15**	**73.46**
DOT-M	1	76.40	20.00	81.89	78.26	**74.01**	89.54	66.99
	2	73.81	20.62	88.89	76.38	70.03	88.34	68.92
	3	73.14	23.08	88.41	78.33	72.48	**91.71**	70.94
	4	**78.11**	**23.30**	**92.20**	**78.84**	72.78	90.81	**71.01**
DOT-P	1	50.00	57.14	00.00	78.69	68.75	90.27	72.90
	2	**66.67**	55.17	00.00	75.86	66.67	88.90	**74.73**
	3	50.00	**64.29**	00.00	**85.25**	**75.00**	**93.76**	72.04
	4	**40.00**	52.17	40.00	82.54	70.83	90.65	72.06
DOT-S	1	72.73	**59.26**	00.00	**33.33**	60.00	76.97	**60.24**
	2	73.47	56.25	00.00	26.67	58.00	82.03	59.79
	3	78.43	23.08	00.00	26.67	50.00	76.92	58.62
	4	**80.00**	52.94	00.00	28.57	**62.00**	**83.71**	58.89
MDT-A	1	82.20	53.64	95.73	84.06	80.37	93.69	73.61
	2	**87.22**	**61.92**	95.20	**88.17**	**84.39**	96.41	**76.44**
	3	84.33	55.63	94.37	86.61	81.71	95.19	75.36
	4	85.40	56.75	**96.07**	85.97	82.07	**96.72**	76.43
MDT-H	1	84.42	54.44	**98.80**	84.37	79.49	93.46	73.62
	2	**86.73**	**65.14**	95.00	**89.09**	**83.80**	96.06	76.33
	3	85.00	58.33	96.30	86.80	81.27	95.24	75.44
	4	85.86	63.58	95.00	87.91	82.78	**96.82**	**76.52**
MDT-M	1	81.82	30.00	96.60	83.27	82.57	95.21	73.30
	2	**88.32**	**36.36**	95.95	87.55	**86.24**	97.16	75.92
	3	84.38	32.14	96.55	**88.12**	84.10	95.77	74.95
	4	86.01	32.65	**97.96**	86.79	85.02	**97.37**	**76.12**
MDT-P	1	50.00	**72.73**	00.00	91.18	**83.33**	93.54	74.56
	2	**66.67**	66.67	00.00	**92.31**	**83.33**	**94.86**	**76.25**
	3	33.33	53.85	00.00	87.10	72.92	92.23	73.35
	4	50.00	42.86	00.00	77.42	64.58	93.26	68.60
MDT-S	1	77.78	66.67	00.00	**57.14**	70.00	85.85	62.86
	2	**87.27**	73.68	00.00	28.57	**78.00**	**94.63**	66.48
	3	**87.27**	**82.35**	00.00	25.00	**78.00**	93.53	**66.70**
	4	84.21	66.67	00.00	00.00	74.00	93.55	66.52

Table 8: F1-scores for text-oriented evaluation. Training sets for evaluation types (T) are the same as in Table 6 rows 1-9. Classifiers: (1) logistic regression (fastText), (2) BiLSTM on word embeddings only (3) BiLSTMd – word embeddings extended using polarity dictionary (4) BERT. Evaluation types are explained in Section 5.

T	C	SP	AMB	0	SN	F1	micro	macro
SDS-H	1	71.98	40.00	64.49	75.90	68.21	83.48	64.44
	2	**82.51**	53.93	72.23	**84.29**	78.31	**93.78**	**73.40**
	3	81.69	51.41	71.21	84.21	77.99	93.43	73.03
	4	82.46	**56.65**	**75.33**	84.21	**78.99**	92.97	72.98
SDS-M	1	67.58	25.90	73.33	64.06	66.18	82.41	61.67
	2	72.36	**31.75**	78.20	71.17	71.96	90.67	70.09
	3	74.49	29.13	79.62	72.58	73.33	**91.18**	**70.39**
	4	**75.69**	27.24	**81.33**	**73.77**	**74.53**	90.76	69.72
SDS-P	1	62.22	**35.34**	33.93	73.19	60.78	80.13	59.96
	2	62.21	28.34	**40.65**	**74.48**	60.78	81.82	61.34
	3	**66.67**	31.46	36.36	73.94	61.32	**83.05**	**62.51**
	4	**66.67**	16.77	36.04	74.07	**62.80**	82.63	60.82
SDS-S	1	**59.34**	58.37	34.29	**42.50**	**54.55**	**77.34**	**59.64**
	2	47.06	47.85	34.29	28.26	43.08	68.40	53.11
	3	45.16	51.61	**35.56**	26.97	43.87	73.38	56.71
	4	51.31	**63.24**	18.18	00.00	51.78	76.17	52.96
DOS-H	1	61.49	26.94	46.98	62.32	54.53	74.29	57.88
	2	72.57	34.60	58.97	74.56	66.56	87.02	67.76
	3	**72.76**	**42.29**	60.50	74.80	**67.81**	**87.89**	**68.21**
	4	70.42	42.12	**60.89**	**74.81**	66.96	85.71	68.07
DOS-M	1	48.58	21.18	56.83	55.56	50.33	71.50	55.83
	2	**61.87**	26.37	62.44	**64.55**	59.47	80.72	63.67
	3	58.68	24.77	63.00	63.00	58.41	80.83	63.51
	4	**61.87**	**27.21**	**66.58**	64.25	**60.75**	**81.80**	**65.08**
DOS-P	1	54.21	23.77	28.92	58.81	47.04	69.03	53.20
	2	66.28	33.33	**35.34**	72.20	**59.30**	**81.78**	**63.82**
	3	**66.47**	30.61	31.50	72.05	58.36	81.15	62.98
	4	64.26	**35.82**	30.95	**72.78**	58.76	78.58	62.11
DOS-S	1	38.52	42.05	**34.92**	30.30	37.15	59.92	52.56
	2	53.25	43.90	19.35	**46.03**	44.27	71.52	58.91
	3	**58.82**	47.50	23.73	41.79	46.64	71.10	**61.07**
	4	55.13	**51.89**	29.79	44.07	**49.01**	**73.09**	59.20
MDS-A	1	66.17	32.36	63.05	66.73	61.27	79.33	61.45
	2	**77.43**	47.21	74.09	**79.40**	74.13	91.48	71.70
	3	77.10	45.88	74.30	78.73	73.70	**91.52**	**71.83**
	4	76.65	**47.76**	**76.70**	79.27	**74.36**	91.19	71.80
MDS-H	1	72.09	33.13	61.42	72.88	65.43	81.43	62.66
	2	**82.82**	51.63	73.18	84.23	78.51	**93.64**	73.19
	3	81.73	54.51	72.68	**84.77**	78.59	93.80	**73.53**
	4	**82.82**	**55.41**	**74.76**	84.52	**78.91**	93.04	73.12
MDS-M	1	63.02	23.12	68.42	61.87	61.37	79.79	60.19
	2	**76.10**	34.88	79.19	75.27	74.44	91.55	70.72
	3	75.27	35.29	79.60	72.51	73.42	91.21	70.72
	4	75.12	**40.00**	**81.83**	**75.50**	**75.67**	**91.71**	**71.52**
MDS-P	1	56.89	31.85	31.75	63.39	52.16	73.92	56.03
	2	67.75	**36.44**	35.93	76.90	63.88	86.03	65.86
	3	**70.65**	35.34	40.00	**77.89**	**65.23**	**87.23**	**67.14**
	4	65.19	33.33	**42.60**	75.53	62.26	84.60	65.06
MDS-S	1	52.17	48.68	26.67	41.44	46.25	69.03	54.72
	2	59.17	**64.42**	34.15	**54.55**	**58.50**	79.16	**62.17**
	3	**61.71**	50.81	30.43	52.00	52.96	78.05	62.10
	4	58.62	53.47	**34.29**	50.53	53.36	**81.38**	61.85

Table 9: F1-scores for sentence-oriented evaluation. Training sets for evaluation types (T) are the same as in Table 6 rows 1-9. Classifiers: (1) logistic regression (fastText), (2) BiLSTM on word embeddings only (3) BiLSTMd – word embeddings extended using polarity dictionary (4) BERT. Evaluation types are explained in Section 5.

Kia Dashtipour, Soujanya Poria, Amir Hussain, Erik Cambria, Ahmad YA Hawalah, Alexander Gelbukh, and Qiang Zhou. 2016. Multilingual sentiment analysis: state of the art and independent comparison of techniques. *Cognitive computation*, 8(4):757–771.

Lingjia Deng and Janyce Wiebe. 2015. Mpqa 3.0: An entity/event-level sentiment corpus. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1323–1328.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Murthy Ganapathibhotla and Bing Liu. 2008. Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 241–248. Association for Computational Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520.

Emitza Guzman and Walid Maalej. 2014. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)*, pages 153–162. IEEE.

Ivan Habernal and Tomáš Brychcín. 2013. Unsupervised improving of sentiment analysis using global target context. In *Proceedings of RANLP 2013*. Association for Computational Linguistics.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee.

George Hripcsak and Adam S. Rothschild. 2005. Technical Brief: Agreement, the F-Measure, and Reliability in Information Retrieval. *JAMIA*, 12(3):296–298.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Arkadiusz Janz, Jan Kocoń, Maciej Piasecki, and Monika Zaśko-Zielińska. 2017. plWordNet as a Basis for Large Emotive Lexicons of Polish. In *LTC'17 8th Language and Technology Conference*, Poznań, Poland. Fundacja Uniwersytetu im. Adama Mickiewicza w Poznaniu.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

Hiroshi Kanayama and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 355–363. Association for Computational Linguistics.

Jan Kocoń and Michał Gawor. 2018. Evaluating KGR10 Polish word embeddings in the recognition of temporal expressions using BiLSTM-CRF. *Schedae Informaticae*, 27.

Jan Kocoń, Arkadiusz Janz, and Maciej Piasecki. 2018a. Classifier-based Polarity Propagation in a Wordnet. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC'18)*.

Jan Kocoń, Arkadiusz Janz, and Maciej Piasecki. 2018b. Context-sensitive Sentiment Propagation in WordNet. In *Proceedings of the 9th International Global Wordnet Conference (GWC'18)*.

Jan Kocoń, Monika Zaśko-Zielińska, and Piotr Miłkowski. 2019. Multi-Level Analysis and Recognition of the Text Sentiment on the Example of Consumer Opinions. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2019*.

Jan Kocoń, Arkadiusz Janz, Miłkowski Piotr, Monika Riegel, Małgorzata Wierzba, Artur Marchewka, Agnieszka Czoska, Damian Grimling, Barbara Konat, Konrad Juszczyk, Katarzyna Klessa, and Maciej Piasecki. 2019a. Recognition of emotions, polarity and arousal in large-scale multi-domain text reviews. In Zygmunt Vetulani and Patrick Paroubek, editors, *Human Language Technologies as a Challenge for Computer Science and Linguistics*, pages 274–280. Wydawnictwo Nauka i Innowacje, Poznań, Poland.

Jan Kocoń, Arkadiusz Janz, Monika Riegel, Małgorzata Wierzba, Artur Marchewka, Agnieszka Czoska, Damian Grimling, Barbara Konat, Konrad Juszczyk, Katarzyna Klessa, and Maciej Piasecki. 2019b. Propagation of emotions, arousal and polarity in WordNet using Heterogeneous Structured Synset Embeddings. In *Proceedings of the 10th International Global Wordnet Conference (GWC'19)*, Wrocław, Poland.

Jan Kocoń and Michał Marcińczuk. 2016. Generating of Events Dictionaries from Polish WordNet for the Recognition of Events in Polish Documents. In *Text, Speech and Dialogue, Proceedings of the 19th International Conference TSD 2016*, volume 9924 of *Lecture Notes in Artificial Intelligence*, Brno, Czech Republic. Springer.

Klaus Krippendorff. 2018. *Content analysis: An introduction to its methodology*. Sage publications.

Siaw Ling Lo, Erik Cambria, Raymond Chiong, and David Cornforth. 2017. Multilingual sentiment analysis: from formal to informal and scarce resource languages. *Artificial Intelligence Review*, 48(4):499–527.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Michal Marcińczuk, Monika Zaśko-Zielińska, and Maciej Piasecki. 2011. Structure annotation in the Polish corpus of suicide notes. In *Text, Speech and Dialogue - 14th International Conference, TSD 2011, Pilsen, Czech Republic, September 1-5, 2011. Proceedings*, volume 6836 of *Lecture Notes in Computer Science*, pages 419–426. Springer.

Michał Marcińczuk, Jan Kocoń, and Bartosz Broda. 2012. Inforex – a web-based tool for text corpus management and semantic annotation. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Michał Marcińczuk and Marcin Oleksy. 2019. Inforex —a Collaborative System for Text Corpora Annotation and Analysis Goes Open. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2019*.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th international workshop on semantic evaluation (semeval-2016)*, pages 1–18.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Maciej Piasecki. 2014. User-driven language technology infrastructure—the case of clarin-pl. In *Proceedings of the Ninth Language Technologies Conference. Ljubljana, Slovenia*.

Maciej Piasecki, Marek Maziarz, Stanisław Szpakowicz, and Ewa Rudnicka. 2014. PLWordNet as the Cornerstone of a Toolkit of Lexico-semantic Resources. In *Proc. 7th International Global Wordnet Conference*, pages 304–312.

Maciej Piasecki, Ksenia Młynarczyk, and Jan Kocoń. 2017. Recognition of genuine Polish suicide notes. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 583–591.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, page 8.

Adam Radziszewski and Tomasz Śniatowski. 2011. Maca — a configurable tool to integrate Polish morphological data. In *Proceedings of the Second International Workshop on Free/Open-Source Rule-Based Machine Translation*.

Jyoti Ramteke, Samarth Shah, Darshan Godhia, and Aadil Shaikh. 2016. Election result prediction using twitter sentiment analysis. In *2016 international conference on inventive computation technologies (ICICT)*, volume 1, pages 1–5. IEEE.

Monika Riegel, Małgorzata Wierzba, Marek Wypych, Łukasz Żurawski, Katarzyna Jednoróg, Anna Grabowska, and Artur Marchewka. 2015. Nencki affective word list (nawl): The cultural adaptation of the berlin affective word list–reloaded (bawl-r) for polish. *Behavior Research Methods*, 47(4):1222–1236.

Anna Rogers, Alexey Romanov, Anna Rumshisky, Svitlana Volkova, Mikhail Gronas, and Alex Gribov. 2018. Rusentiment: An enriched sentiment analysis dataset for social media in russian. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 755–763.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

V Subramaniyaswamy, R Logesh, M Abejith, Sunil Umasankar, and A Umamakeswari. 2017. Sentiment analysis of tweets for estimating criticality and security of events. *Journal of Organizational and End User Computing (JOEUC)*, 29(4):51–71.

Maite Taboada, Kimberly Voll, and Julian Brooke. 2008. Extracting sentiment as a function of discourse structure and topicality. *Simon Fraser Univeristy School of Computing Science Technical Report*.

Marek Troszyński and Aleksandra Wawer. 2017. Czy komputer rozpozna hejtera? wykorzystanie uczenia maszynowego (ml) w jakościowej analizie danych. *Przegląd Socjologii Jakościowej*, 13(2):62–80.

Fei Wang, Yunfang Wu, and Likun Qiu. 2012. Exploiting discourse relations for sentiment analysis. *Proceedings of COLING 2012: Posters*, pages 1311–1320.

Aleksander Wawer. 2012. Mining co-occurrence matrices for so-pmi paradigm word candidates. In *Proceedings of the Student Research Workshop at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 74–80. Association for Computational Linguistics.

Aleksander Wawer and Maciej Ogrodniczuk. 2017. Results of the poleval 2017 competition: sentiment analysis shared task. In *8th Language and Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics*.

Aleksander Wawer and Dominika Rogozinska. 2012. How much supervision? corpus-based lexeme sentiment estimation. In *2012 IEEE 12th International Conference on Data Mining Workshops*, pages 724–730. IEEE.

Małgorzata Wierzba, Monika Riegel, Marek Wypych, Katarzyna Jednoróg, Paweł Turnau, Anna Grabowska, and Artur Marchewka. 2015. Basic emotions in the nencki affective word list (nawl be): New method of classifying emotional stimuli. *PLoS One*, 10(7):e0132305.

Monika Zaśko-Zielińska. 2013. *Listy pożegnalne: w poszukiwaniu lingwistycznych wyznaczników autentyczności tekstu*. Wydawnictwo Quaestio, Wrocław.

Monika Zaśko-Zielińska, Maciej Piasecki, and Stan Szpakowicz. 2015. A large wordnet-based sentiment lexicon for polish. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 721–730.

Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253.

Wenbin Zhang and Steven Skiena. 2010. Trading strategies to exploit blog and news sentiment. In *Fourth international aAAI conference on weblogs and social media*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 207–212.

A Personalized Sentiment Model with Textual and Contextual Information

Siwen Guo **Sviatlana Höhn** **Christoph Schommer**

ILIAS Research Lab, CSC,
University of Luxembourg
{siwen.guo,sviatlana.hoehn,christoph.schommer}@uni.lu

Abstract

In this paper, we look beyond the traditional population-level sentiment modeling and consider the individuality in a person's expressions by discovering both textual and contextual information. In particular, we construct a hierarchical neural network that leverages valuable information from a person's past expressions, and offer a better understanding of the sentiment from the expresser's perspective. Additionally, we investigate how a person's sentiment changes over time so that recent incidents or opinions may have more effect on the person's current sentiment than the old ones. Psychological studies have also shown that individual variation exists in how easily people change their sentiments. In order to model such traits, we develop a modified attention mechanism with Hawkes process applied on top of a recurrent network for a user-specific design. Implemented with automatically labeled Twitter data, the proposed model has shown positive results employing different input formulations for representing the concerned information.

1 Introduction

Sentiment is one of the key factors affecting human behavior. Studying the way in which sentiment is perceived, evolved and expressed is an essential part in artificial intelligence. To analyze sentiment in text, researchers have made different assumptions on linguistic behaviors that are leveraged with approaches developed based on the nature of the text, the representation of the related information and the objectives. However, majority of the studies are conducted at the population-level which assumes that people follow a common understanding with regard to the use of language. Such approaches can be inaccurate in the cases where people use the same lexical choices to convey different messages or vice versa. Harris (2006) stated that 'no two alike' showing the inherent difference in human that motivates the research of personalized sentiment analysis. Grounded in the psychological works, we argue that it is significant to study the effect of individuality in the expressions and to investigate the possibility of providing a deeper understanding of the expressions from the writers' own perspectives. In this work, we concern the use of preferred lexical choices when expressing sentiment (Reiter and Sripada, 2002) and the level of consistency in retaining a sentiment (Janis and Field, 1956).

Besides the targeted text message itself, we exploit two types of contextual information for the purpose of realizing the psychological aspects: a person's expressions in the past and the time when the expressions were made. With the goal of discovering the effect of the contextual information, distinct formulation methods are proposed to integrate the information in the personalized sentiment model. The backboned model is a hierarchical neural network which follows a conventional embedding – recurrent – attention structure with each part rectified for the task. The embedding block is used to generate representations for the used information; the recurrent network fulfills the task of relating to the information from the past; the attention model is shaped with Hawkes process (Laub et al., 2015) in order to model the information decay for each expresser. Generally, recurrent networks consider the order of the elements in a sequence but omit the different gaps between them. Hawkes process is utilized to compensate this issue. Furthermore, a novel approach with a user – factor transformation is employed to merge the Hawkes process within the attention model and to construct user-specific processes. For evaluation, we take the data from a number of frequent users on social platforms where Twitter is used as an example. The data is domain-independent, and

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 992–1001
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

it is possible to obtain self-labeled texts that aligns with our goal of understanding the expressers' perspectives. Significant improvements are seen with certain input formulations, and different Hawkes processes applied for the users are visualized. In the end, we conclude that it is effective to introduce the contextual information to the model.

2 Related Work

Individualities are mostly considered in sentiment analysis when analyzing product-review texts (Gong et al., 2016; Chen et al., 2016b; Wu et al., 2018). A common issue that challenges the research of this area is data sparsity. It is infeasible to build or train an effective model for each user. Gong et al. (2016) address this issue by relating to a global model that captures 'social norms', and individualities are included by adapting from the global model via a series of linear transformations. While in the works that apply neural networks, the user information is embedded separately (Chen et al., 2016b) or added at the attention layer (Chen et al., 2016a; Wu et al., 2018) in order to make user-specific predictions at the output. In other works, the aspect of personalization is relaxed to provide user-group based predictions (Gong et al., 2017). The aforementioned approaches are modeled with domain-dependent text, while our concentration on the text associated with various topics makes the task more challenging. Wu and Huang (2016) focus on microblog posts as well and apply the same concept of using a global model and an individual model via multi-task learning as in Gong et al. (2016). In addition, users' social relations are leveraged to enhance the individual models. Similarly, followers' and followees' information is also used in Song et al. (2015) while a variant of latent factor model is utilized. Most of the studies leverage earlier posts from users in order to better understand the individuality; however the evolvement of the sentiment is largely neglected — the preferences of users are considered constant. In this work, we take the user dynamics into consideration, and incorporate user information both in the input and in the Hawkes process to deal with the data sparsity and to offer personalized analysis.

Technically, there are very few works that investigate the different gaps between the input nodes in a recurrent neural network. Neil et al. (2016) have proposed a phased LSTM that utilizes an additional time gate to control the passing of the information. However, the time gate is triggered by periodic oscillations while modeling sensory events, which makes such a design less flexible when the time gaps are highly various. As an alternative, we explicitly add the time gaps in the Hawkes process to offer a time-sensitive modeling.

In our previous works, we have evaluated the effectiveness of considering individual differences in sentiment analysis by employing a concept-based representation and the static or universal Hawkes process (Guo et al., 2018, 2019a). In this paper, we advance the development by adopting five input formulations with different combinations of granular levels, and propose a refined model with user-specific Hawkes process to constitute a step forward in capturing the nuances of user dynamics and providing insights in the personalized modeling.

3 Personalized Sentiment Model

Motivated by the diversity in individuality, we introduce a model that considers both textual and contextual information and applies hierarchical neural networks to facilitate the aspect of personalization in sentiment analysis. Particularly, we focus on analyzing the effect of contextual information and discover ways to embed such information in the prediction process.

3.1 Model Structure

The personalized sentiment model follows a conventional embedding – recurrent – attention structure as shown in Figure 1 with modifications applied at each block. First, a formulation method is applied in order to represent the current and a number of earlier posts of a user. The embedding

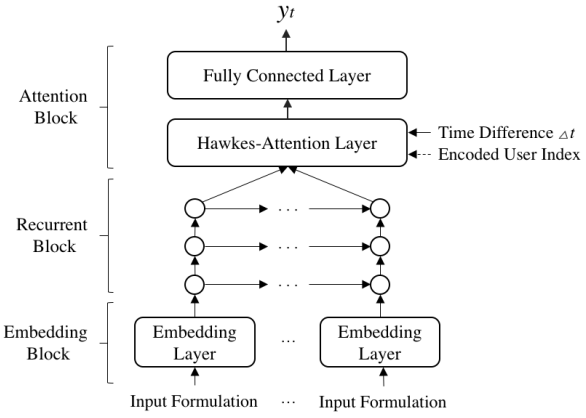


Figure 1: The structure of the sentiment model.

layer takes the formulated inputs and produces a vector for each time step at the recurrent layer. After that, the outputs of the recurrent layer together with the auxiliary time differences are fed to the attention block. Encoded user index is used in the Hawkes-attention layer when different settings of Hawkes process are considered for different users. A fully connected layer is applied afterwards to regularize the output of the Hawkes-attention layer. Finally, the output of the model y_t is generated which is the predicted sentiment label of the target text.

3.2 Textual and Contextual Information

In linguistic studies, the notion of 'context' varies from theory to theory that some in monologism see it as 'secondary complications', whereas in dialogist theory, it considers the reflexive relation between an expression and its setting or occasion essential (Linell, 2009). In this work, we target social text and regard context as an important factor in the setting of social platforms. Based on the characteristics of such text and the applicability in modeling, we categorize the information used in the sentiment model into two genres: **Textual Information** — the information that can be extracted directly from the *target* text, and **Contextual Information** — the information that is *not* present *in* the target text but is *associated* with the text according to the user index.

Textual Information

Text is the central part in sentiment analysis. Researchers in this area have proposed various approaches aiming to provide a deep text understanding given the complex nature of how people express their sentiments in text. Moreover, methods designed at the population-level for many other natural language processing tasks can also be used for sentiment analysis. Generally, such methods start at a pre-defined granular level and generate a representation for the text by capturing related information in each granule and the ones surrounding it. In the end, the text is represented explicitly (e.g., concepts as in Poria et al., 2014) and / or implicitly (e.g., embeddings as in Pennington et al., 2014 and Peters et al., 2018).

Contextual Information

Besides the *target* text itself, other types of information have been used to support the understanding of the sentiment as well.

Earlier Posts correspond to the texts produced by the same user in the past. The use of earlier posts leverages the assumption that a person may have similar lexical choices when expressing opinions about related topics while different individuals share different preferences in this regard. By analyzing the lexical choices and the topics or entities associated with them, the tendency of repeating such patterns in a user's text in the future can be beneficial to the prediction.

Timestamp corresponds to the creation time of the text. It has been shown that there exists a certain level of consistency in an individual's opinions and such consistency varies from one individual to another (Janis and Field, 1956). We study such a trait by taking the timestamp of each earlier post and applying Hawkes process to observe how the effect of the information on a user's behaviors or opinions decays over time. Note that here, we do not distinguish the inconsistency between the time when the expression was made and the time when the sentiment was felt.

3.3 Input Formulations

We employ different formulations for the input sequence based on the representation method. For all the formulations, timestamps are apart from other information and are used as an auxiliary input directly at the attention layer with Hawkes process. Additionally, user index is used as a feature in the input in order to handle the issue of data sparsity, and by doing that, the model is able to analyze textual and contextual relations targeting a specific user. The encoded user index is also used at the Hawkes-attention layer when considering individual differences in information decay.

Atomic Representation (AR)

In this formulation, four types of components are extracted: concepts, entities, negations and user index. Concepts are extracted based on Cambria et al. (2018) which contain conceptual and affective information, and can be seen as the 'signal terms' regarding lexical choices. Entities are extracted based on grammatical rules as the 'targets' of a user's lexical choice. Negations are extracted based on the lexicon by Reitan et al. (2015) for their ability to invert the orientation of a sentiment. User index is extracted for its role in personalization. After extracting the components, an embedding layer is applied to generate a representation vector for the text.

Representation with Pre-trained Word Embeddings (WE)

Pre-trained word vectors such as GloVe (Pennington et al., 2014) and Word2Vec (Mikolov et al., 2013), generate embeddings according to the co-occurrences of the words. The word embeddings are aggregated dimension-wise to produce a vector for each post. The user index is encoded by itself and then combined with the representation of the post at each time point of an input sequence.

Representation with Concepts and Words (CW)

Since the pre-trained word vectors do not consider the contexts of the target words, we combine the representations of both words and concepts in this formulation. The word embeddings are taken as the same as the one in the **WE** formulation. The concepts appeared in the text are encoded together with the associated user index so that the relation between the user and the use of concepts can be learned. Afterwards, the two types of representation of the same post are concatenated to generate an input sequence for the recurrent layer.

Representation with Deep Contextualization (DC)

Peters et al. (2018) proposed a deep contextualized representation (ELMo) that takes a finer granular level (characters) to generate embeddings for the text by leveraging a deep bidirectional language model. Prominent results are shown across a number of linguistic tasks using this representation. We apply ELMo for each post, and then combine the representation of the post and the encoded user index at each time point.

Representation with Combined Granular Levels (Combi)

This formulation combines three granular levels, namely character-level (**DC**), word-level and concept-level (**CW**). The representations are embedded separately and are concatenated afterwards as mentioned in Peters et al. (2018).

3.4 Recurrent Neural Network with Input Selection from Post History

We apply a deep recurrent neural network with long short-term memory (LSTM, Hochreiter and Schmidhuber, 1997) on the input sequences constructed with one of the input formulations. Each input sequence X_i consists of an entry of the current post at the end of the sequence (which contains textual information and the encoded user index) and a number of earlier posts by the same user (contextual information), i.e.,

$$X_i = [H_{i-n}, ..., H_{i-2}, H_{i-1}, F_a(x_i)] \quad (1)$$

where

$$H_j = \begin{cases} F_a(x_j) & \text{if } u(x_j) = u(x_i) \\ 0 & \text{else} \end{cases},$$

n is the number of earlier posts considered, F_a is the formulation chosen beforehand, and u is the user index of the post.

Additionally, a selection procedure followed Guo et al. (2019b), is performed for choosing the earlier post x_j of a target text x_i from user u. The use of this procedure is motivated by the large difference in user frequency (the number of posts of a user in a given corpus), as well as the observation of the case where the recent posts are unrelated to the current one while the related posts have appeared long before. The selection is done by calculating the similarity between the topics of each earlier post and the target text. Given a fixed number of time steps T in a recurrent network and a similarity threshold δ, the recent T earlier posts that have a similarity score larger or equal to δ are chosen. For the case where the number of chosen posts is smaller than T, other earlier posts are added in the sequence as complements prioritizing on the recent ones. After the selection, the posts in each sequence are ordered by time.

3.5 Hawkes-Attention Layer

A modified attention mechanism is applied on top of the recurrent neural network. Attention model has the ability to provide more flexibilities at the output layer (Bahdanau et al., 2015). The network can 'attend' to different histories based on the immediate situation. As in Yang et al. (2016), the model is defined as follows:

$$u_i = \tanh\left(W_t h_i + b_t\right) \quad (2)$$

$$\alpha_i = \frac{exp(u_i^\top u_t)}{\sum_i exp(u_i^\top u_t)} \quad (3)$$

$$\lambda_i = \alpha_i h_i \quad (4)$$

$$v = \sum_i \lambda_i \quad (5)$$

where h_i is the i-th output of the recurrent network, u_i is the hidden representation of h_i, and u_t is the 'context vector'. Here, we randomly initialize the context vector which is later jointly learned with other weights during the training phase. λ_i is the representation of the information learned at time step i. Lastly, v summarizes all the information of the posts from the corresponding sequence. However, conventional recurrent networks and attention models do not differentiate the relations between time steps regarding various time intervals. To model this difference, we shape the representation of the post λ_i with Hawkes process before summarizing them at the last step (Equation 5) in the attention mechanism.

Universal Hawkes Process

Hawkes process is known for modeling the excitation and decay of information over time. When using exponential decay as the excitation function, the Hawkes conditional intensity is (Laub et al., 2015):

$$\lambda^*(t) = \lambda + \sum_{t_i<t} \alpha e^{-\beta(t-t_i)} \tag{6}$$

where λ describes the positive background intensity, t is the current time and t_i is the time when the past event happened. α and β are the most important factors in the Hawkes process that the former corresponds to the amount of excitement the past event brought to the system while the latter corresponds to the decay rate of the excitement. Taking the same concept as in Guo et al. (2019a), we see a post in the past as an 'event' that can influence the decision in the future and such influence decays over time. Instead of treating all the past events equally, we use λ_i in Equation 4 as the background intensity and $\alpha = \epsilon\lambda'_i$ as the amount of excitement the post at time step i contributes to the current decision. Note that $\lambda'_i = max(\lambda_i, 0)$ for we do not consider negative effect from the past. With this modification, the information decay of a past event can also depend on the relativeness between the past and the current events, and ϵ can be seen as a scaler to balance the importance of adding the process. As a result, Equation 5 is replaced with the following:

$$v' = \sum_{i:\Delta t_i \geqslant 0} (\lambda_i + \epsilon\lambda'_i e^{-\beta\Delta t_i}) \tag{7}$$

where Δt_i indicates the time gap between the earlier post at time step i and the current post. The current post is included in the summarization when $\Delta t_i = 0$. ϵ and β are learned jointly with other learnable parameters during the training phase.

User-specific Hawkes Process

In order to build user-specific Hawkes process, we compute the values of ϵ and β in Equation 7 for each user by applying learned transformation vectors on the encoded user index. In this way, different behaviors concerning the information decay can be analyzed. That is, ϵ and β are calculated as

$$\epsilon = a_\epsilon^\top E(u) \tag{8}$$

$$\beta = a_\beta^\top E(u) \tag{9}$$

where E is the user-index encoder. The transformation vectors a_ϵ and a_β are learned during the training process, and other settings remain the same with the universal Hawkes process.

Similarly, Cao et al. (2017) also integrate a Hawkes process in a neural-based system. To avoid pre-defining a time decay function, the time range in an observation is split into a number of disjoint intervals, whereas user information is embedded in the input. Although this non-parametric method can be applied in our model, the selection of the number of intervals undermines the flexibility of the process. However, as in their work, a fully connected layer is applied afterwards which takes the exited (decayed) information representation v' as input and outputs the final prediction of the sentiment y_t.

4 Experiments

We investigate the effect of textual and contextual information in personalization and evaluate the performance of the model employing different input formulations.

4.1 Dataset

The Sentiment140[1] corpus is chosen in the experiments for complying the requirements that

1. there are sufficient frequent users,
2. the text is domain-independent,
3. the desired textual and contextual information is present,

[1] http://help.sentiment140.com/for-students, last seen on September 24, 2019

4. the corpus is annotated from the writers' perspectives.

The corpus is labeled automatically by emoticons as described in Go et al. (2009) and reflects a user-specific view in contrast to the corpus labeled by others such as the SemEval[2] corpus. However, the automatic labeling may also contain a certain level of noise caused by the variation in emoticon usage and the unreliability at the user end. The experimented dataset is created by taking the messages from the users who have posted at least 20 times before a pre-set timestamp. This results in 2369 users with overall 122,000 messages in which 79,009 are positive and 42,991 are negative. Furthermore, the dataset is split into a training set, a development set and a test set according to two pre-set time points to ensure that the prediction is only made based on the messages in the past. Other details of the dataset can be found in the appendix.

4.2 Experimental Settings

The experiments are conducted using Keras[3] with TensorFlow[4] backend. The concepts used in the **AR** and **CW** formulations are based on SenticNet 5[5]. In **WE** and **CW**, 100 dimensional Twitter word vectors are taken from GloVe[6]. The ELMo word representations in the **DC** input formulation are supported by TensorFlow Hub[7], which are later re-trained with other weights in the model. The inputs with **AR**, **WE** and **DC** are encoded into different lengths based on the number of elements in each formulation, however they are suppressed at the embedding layer that generates a vector of length 100 at each time step in order to make fair comparisons. In **CW** and **Combi**, the input vectors fed to the recurrent layer are longer (164 and 264 respectively) because of the concatenation of representations. The dimension of user embeddings is set to 32. There are three recurrent layers at the recurrent block that each contains 100 units, and the number of time steps T is set to 20. For the selection procedure, the same setting is used as in Guo et al. (2019b), where Manhattan distance is used as the ground measurement for calculating topic similarities and the similarity threshold δ is set empirically at 0.8. At the attention block, the time unit is hour, and the values of ϵ and β are initialized at 0.01 and 0.001 respectively when using the universal Hawkes process; the initial values for the vectors a_ϵ and a_β when using user-specific processes are also vectors of 0.01 and 0.001 respectively, and the length of the vectors has to be the same with the dimension of user embeddings, which is 32. The dimension of the fully connected layer applied before the output is set to the same as the number of units in the recurrent layer. We report the overall accuracy of the model as well as the F_1 scores for the positive and negative classes. Detailed settings of the model, sample codes for the Hawkes process, and trained models with the **Combi** formulation can be found in the supplementary material.

4.3 Results

Table 1 shows the performance of the model in different settings. The best result is given by the **Combi** formulation with user-specific Hawkes process. Comparing to the result we have reported previously in Guo et al. (2019a) with an accuracy of 76.13, the best performance with this model has reached 80.38 using the same test set.

Results with Different Input Formulations

Across different input formulations, improvements can be seen comparing the models using the universal - and the user-specific Hawkes process. Although the increase when applying the **AR** formulation is not significant, the improvement of other formulations are significant (t test with $p < 0.05$). The lack of improvements with the **AR** formulation when learning user-specific behaviors can be caused by the sparser representation compared to the other formulations. A matching from each post to a list of concepts and negations is performed which omits information that is not present in the given list. The list of concepts provided by SenticNet 5 is more restricted than the word vectors by GloVe, and is far less flexible than the character-based representation. In addition, due to the highly unstructured nature of social texts, the preprocessing of the posts plays a significant role in the **AR** formulation, which also affects the per-

[2] http://alt.qcri.org/semeval2017/task4/, last seen on September 24, 2019

[3] https://keras.io/, last seen on September 24, 2019

[4] https://www.tensorflow.org/, last seen on September 24, 2019

[5] https://sentic.net/downloads/, last seen on September 24, 2019

[6] https://nlp.stanford.edu/projects/glove/, last seen on September 24, 2019

[7] https://tfhub.dev/google/elmo/2, last seen on September 24, 2019

Input Formulation	Universal Hawkes Process			User-specific Hawkes Process		
	Pos. F1	Neg. F1	Accuracy	Pos. F1	Neg. F1	Accuracy
AR	75.12	76.87	76.04	74.94	77.48	76.28
WE	76.06	77.06	76.58	76.61	78.41	77.55
CW	76.43	77.71	77.10	76.90	79.10	78.06
DC	76.60	79.71	78.27	77.10	80.36	78.86
Combi	77.78	80.67	79.34	78.06	82.25	80.38

Table 1: Performance of the sentiment model when applying the universal - and the user-specific Hawkes process with different input formulations.

formance substantially.

We can also observe improvements when using finer granular levels which are more sensitive and representative towards user variations. Note that using the character-based **DC** formulation alone offers better performance than using the combination of word - and concept-level representations; however the ELMo representation has a more complex structure, a higher dimensional output, and it takes longer time to re-train the weights in the network. In conclusion, the best solution for constructing representation for the inputs is to leverage the combined granular levels from character to word, and to concept (**Combi**). With such a representation, the system is able to analyze user-specific behaviors regarding the lexical usage and the consistency of sentiment.

Results for Various Lengths of History

Figure 2 shows the performance of the models while using the **CW** formulation. The models are tested for T from 1 where no earlier posts are considered, to 20 after which no significant improvement can be observed due to the number of related posts a user normally publishes.

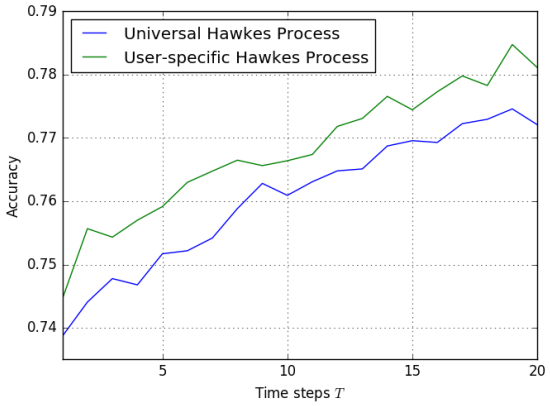


Figure 2: Comparison of the universal - and the user-specific Hawkes process-based models while using different time steps.

For the case when the user history is not incorporated in the model ($T = 1, \Delta t = 0$), we can deduce that $v' = \lambda + \epsilon\lambda'$, which leads to an accuracy of 73.87 with the universal ϵ and 74.46 with the user-specific ϵ (Equation 8).

We can observe an increase in both models when rising the number of time steps T. The increase indicates that the personalization is effective and earlier posts are valid contextual information. By using the selection procedure, the models with a smaller number of T (except for when $T = 1$) take into account more related posts in the past. The increase grows slightly faster towards smaller numbers of T, which is also caused by the limitation of user frequencies in the experimented corpus. We believe that given a sufficient number of frequent users, the performance of the proposed models can be further improved.

Results for Various User Frequencies

The performance of the models when applying for users with different frequencies can be seen in Figure 3. The x-axis corresponds to the lower bound of the user frequency. We take the lower bound for the illustration because there are different numbers of users for each frequency, and many frequencies have no users to assign to. With both models, significant growths for each input formulation can be observed while increasing the lower bound of the frequency. Note that although the **Combi** formulation gives the overall best performance, we can see from the figure that it does not give the best results in all the cases. For instance, when the user frequency is around 80, the **WE** formulation has the best accuracy in both models. However, such an observation is also restricted by the number of frequent users in general — with only 372 posts in the test set when the user frequency is at least 100, the performance is highly dependent on the remaining 3 users. Another observation is that the

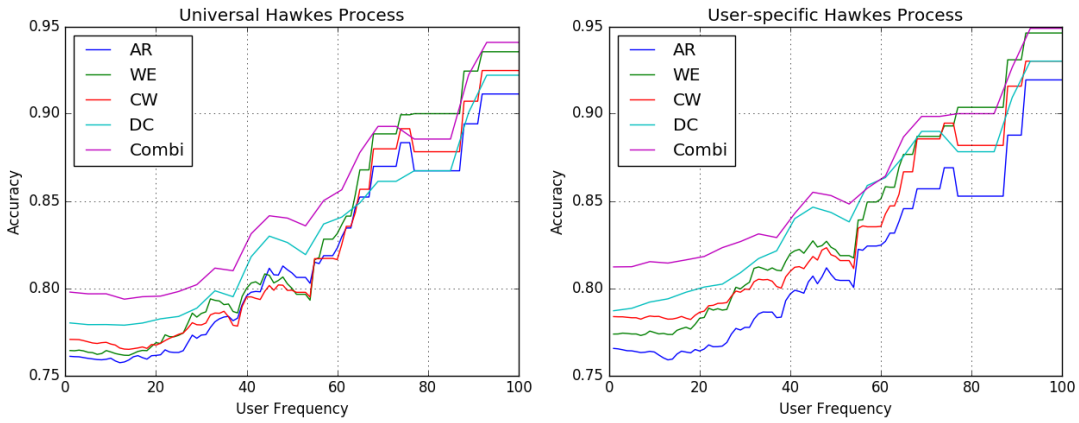


Figure 3: Performance of the universal - and the user-specific Hawkes process-based models for users with different frequencies when applying different formulations.

WE formulation performs better than the **CW** formulation in higher user frequencies, but it has a lower overall performance because there are more users who have published less than 30 posts than the ones who have more.

4.4 Visualization of User-specific Hawkes-Attention

In order to examine the user-specific Hawkes process, we visualize the intermediate calculations for the values of ϵ and β for 10 random users (Figure 4). Each cell in the figure corresponds to the value of $a_{\epsilon i} * E_i(u)$ in Equation 8 (top figure) or $a_{\beta_i} * E_i(u)$ in Equation 9 (bottom figure) at dimension i for the respective user.

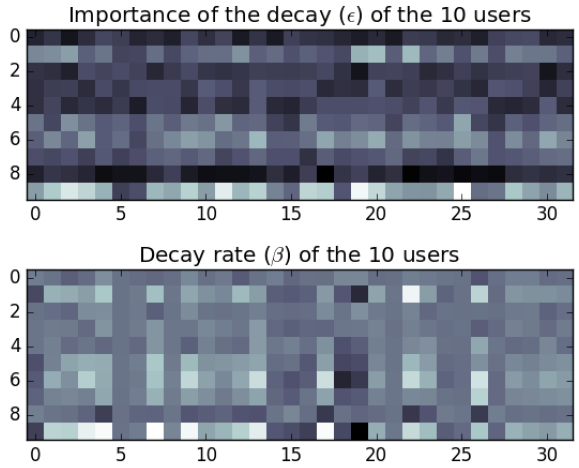


Figure 4: The vectors of ϵ and β in the user-specific Hawkes process of the random 10 users.

The effect of the learned transformation vectors on the 10 users is illustrated. It can be seen that the last user in the figure (the one at the bottom line) has the greatest values for ϵ and β, which means that the decay factor has a great impact on the prediction for this user than the others but the influence from the past decays comparably fast — the user is affected a lot by recent events. In contrast, among the 10 users, the second last user is the least influenced by the past which is visualized in darker colors. From this figure, we can see that the different decaying processes are indeed learned for different users with the vector transformation. One may argue that the behavior of the Hawkes process also depends on the time period of the experimented dataset; however, if an earlier post (outside of the training period) is highly relevant to the current one, the large value of λ_i can still prevail regardless the value of ϵ.

5 Conclusion

This paper presents a personalized sentiment model that captures the individualities in expressing sentiment and analyzes the evolvement of sentiment over time. Particularly, we categorize the information used for the modeling into textual and contextual information, and evaluate the effectiveness of using the contextual information to boost the performance of the model. A novel attention mechanism with user-specific Hawkes process is employed for this purpose. Technically, it also provides an alternative for studying various time gaps in temporal sequences with neural networks. Different input formulations are applied in which the combined granular representation performs the best. Based on our findings, we can conclude that the individual variation indeed affects the analy-

sis, and the contextual information, as an essential part in human interactions, positively contributes to the performance.

Because the informal text we have used deviates from the language standard, the representation of input text plays a significant role in improving the performance. In the future work, we will exploit phonetic representation which can provide another source of information for such text. The posts can be transcribed into phonetic sequences, for instance, by using the International Phonetic Alphabet, in order to handle certain misspellings and to study the trend of using letters with similar pronunciations as substitutions. Moreover, other types of contextual information should be explored as well to enhance the understanding of individual behaviors on social platforms. As an example, social relations can be used to identify abnormalities in the change of sentiment, especially in the case that a user is exceptionally stimulated by other users or special events which causes untypical behaviors. The personalized model can also be helpful in other scenarios, such as to offer deep understanding for user-tailored conversations or companionship.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Erik Cambria, Soujanya Poria, Devamanyu Hazarika, and Kenneth Kwok. 2018. SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. In *Proceedings of AAAI*.

Qi Cao, Huawei Shen, Keting Cen, Wentao Ouyang, and Xueqi Cheng. 2017. DeepHawkes: bridging the gap between prediction and understanding of information cascades. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1149–1158. ACM.

Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. 2016a. Neural sentiment classification with user and product attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1650–1659.

Tao Chen, Ruifeng Xu, Yulan He, Yunqing Xia, and Xuan Wang. 2016b. Learning user and product distributed representations using a sequence model for sentiment analysis. *IEEE Computational Intelligence Magazine*, 11(3):34–44.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Lin Gong, Mohammad Al Boni, and Hongning Wang. 2016. Modeling social norms evolution for personalized sentiment classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 855–865.

Lin Gong, Benjamin Haines, and Hongning Wang. 2017. Clustered model adaption for personalized sentiment analysis. In *Proceedings of the 26th International Conference on World Wide Web*, pages 937–946. International World Wide Web Conferences Steering Committee.

Siwen Guo, Sviatlana Höhn, and Christoph Schommer. 2019a. Looking into the past: evaluating the effect of time gaps in a personalized sentiment model. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1057–1060. ACM.

Siwen Guo, Sviatlana Höhn, and Christoph Schommer. 2019b. Topic-based historical information selection for personalized sentiment analysis. In *Proceedings of the 27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 379–384.

Siwen Guo, Sviatlana Höhn, Feiyu Xu, and Christoph Schommer. 2018. Personalized sentiment analysis and a framework with attention-based Hawkes process model. In *International Conference on Agents and Artificial Intelligence*, pages 202–222. Springer.

Judith Rich Harris. 2006. *No two alike: Human nature and human individuality*. WW Norton & Company.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Irving L Janis and Peter B Field. 1956. A behavioral assessment of persuasibility: Consistency of individual differences. *Sociometry*, 19(4):241–259.

Patrick J Laub, Thomas Taimre, and Philip K Pollett. 2015. Hawkes processes. *arXiv preprint arXiv:1507.02822*.

Per Linell. 2009. *Rethinking language, mind, and world dialogically*, chapter 3. IAP.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. 2016. Phased LSTM: Accelerating recurrent network training for long or event-based sequences. In *Advances in Neural Information Processing Systems*, pages 3882–3890.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Soujanya Poria, Erik Cambria, Grégoire Winterstein, and Guang-Bin Huang. 2014. Sentic patterns: Dependency-based rules for concept-level sentiment analysis. *Knowledge-Based Systems*, 69:45–63.

Johan Reitan, Jørgen Faret, Björn Gambäck, and Lars Bungum. 2015. Negation scope detection for Twitter sentiment analysis. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 99–108.

Ehud Reiter and Somayajulu Sripada. 2002. Human variation and lexical choice. *Computational Linguistics*, 28(4):545–553.

Kaisong Song, Shi Feng, Wei Gao, Daling Wang, Ge Yu, and Kam-Fai Wong. 2015. Personalized sentiment classification based on latent individuality of microblog users. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Fangzhao Wu and Yongfeng Huang. 2016. Personalized microblog sentiment classification via multitask learning. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Zhen Wu, Xin-Yu Dai, Cunyan Yin, Shujian Huang, and Jiajun Chen. 2018. Improving review representations with user attention and product attention for sentiment classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Cluster-gated Convolutional Neural Network for Short Text Classification

Haidong Zhang, Wancheng Ni, Meijing Zhao, Ziqi Lin
Institute of Automation, Chinese Academy of Sciences, China
haidong_zhang14@yahoo.com
{wancheng.ni, meijing.zhao, linziqi2013}@ia.ac.cn

Abstract

Text classification plays a crucial role for understanding natural language in a wide range of applications. Most existing approaches mainly focus on long text classification (e.g., blogs, documents, paragraphs). However, they cannot easily be applied to short text because of its sparsity and lack of context. In this paper, we propose a new model called cluster-gated convolutional neural network (CGCNN), which jointly explores word-level clustering and text classification in an end-to-end manner. Specifically, the proposed model firstly uses a bi-directional long short-term memory to learn word representations. Then, it leverages a soft clustering method to explore their semantic relation with the cluster centers, and takes linear transformation on text representations. It develops a cluster-dependent gated convolutional layer to further control the cluster-dependent feature flows. Experimental results on five commonly used datasets show that our model outperforms state-of-the-art models.

1 Introduction

With the rapid development of social media, e-commerce and on-line communication, the Internet has been generating an increasing amount of short texts, including texts, search snippets, user reviews for products, etc., which poses an urgent demand for understanding them. Short text classification, assigning predefined categories to texts, is a fundamental technique in natural language processing, and plays an important role in a wide range of applications, such as sentiment analysis, web searching, and ads matching.

In prior research, much progress has been made on text classification, including traditional approaches based on human-designed features (Lazaridou et al., 2013; Zhang et al., 2015a) and neural networks based on deep architectures (Lai et al., 2015; Yang et al., 2016). However, such methods prefer to deal with documents and paragraphs, and still have limitations for short texts. Each short text does not have enough words, which may result in data sparsity and lack of contexts (Wang et al., 2017).

Some researchers incorporated knowledge bases into traditional approaches (Feng et al., 2013; Wang et al., 2014) or neural networks (Wang et al., 2017) to overcome these challenges. Extra resources can provide abundant semantic information for short text classification, but the performance of such methods is strongly dependent on the quality of knowledge bases and constructing a large-scale knowledge base is time-consuming and labor-intensive. Another strategy is to explore latent topics (Chen et al., 2011; Ren et al., 2016) or clustering features (Ma et al., 2015; Revanasiddappa and Harish, 2018) for texts and input them into some classifiers as features. Such methods can reduce high dimensionality and terms' sparse distribution problems. Their shortness is to use pre-trained topics or clusters as features, which might be hard to explore the potential association between clustering and classification.

To address the limitations, we construct a joint architecture to embed a soft clustering method into the classification task, because joint architectures can leverage mutual information for each other and have been useful in many studies for understanding natural language (Luo et al., 2015; Shao et al., 2017; Schmitt et al., 2018). In addition, convolutional neural network and the gated mechanisms have been proven effectiveness in sentence-level language modeling (Dauphin et al., 2016; Gehring et al., 2017), and cluster centers of words contain semantic closeness of similar ones, which motivate

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 1002–1011
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

us to utilize them to auto-extract and highlight the cluster-related features for classification.

Based on the above analysis, we propose a joint model called cluster-gated convolutional neural network (CGCNN), coupling clustering and classification methods, to construct an end-to-end deep architecture. It integrates a soft clustering method into a gated convolutional neural network, which can explore the semantic relation of word-level context and the global corpus. And it can also guide the gating mechanism unit to control cluster-dependent feature flows. Specifically, it firstly uses a bi-directional long short-term memory model (BiLSTM) to learn word representations and capture local context in text. Then, it performs a soft clustering method on word representations for the probability of each word assigning to each cluster, which can build a bridge between word and the global corpus. And we develop a linear transformation to calculate cluster-dependent text representations. Based on the gating mechanism, we uses the cluster centers to further highlight the cluster-dependent convolutional features for the corresponding cluster. At last, we perform max-over-time pooling and concatenation operations to combine the selected features for classification.

The main contributions of this study are summarized as follows:

- We develop a joint model that combines clustering and classification methods in an end-to-end manner. The model leverages the semantic relation of words and the global corpus by learning from a soft clustering method to assist the classification task.

- To the best of our knowledge, our model is the first to incorporate a clustering method into the gating mechanism for convolutional neural network, which can help to control related features with clusters.

- We conduct extensive experiments on five real-world datasets to verify the effectiveness of our model. The experiment results show that the proposed method outperforms state-of-the-art methods.

2 Related Work

In this section, we review the related work from the following two aspects: text classification and short text classification.

2.1 Text Classification

Traditional text classification methods generally rely on manual features, such as bag-of-words, short n-grams, POS tagging. Most recent studies design more complex features for specific applications. For example, Lazaridou et al. (2013) considered discourse connectives (such as "*but*", "*and*") in the Bayesian model for sentiment classification. Post and Bergsma (2013) used multiple explicit and implicit syntactic features (e.g., unigrams, bigrams, and grammar tree patterns) for text classification. Zhang et al. (2015a) integrated word embeddings learned by word2vec into support vector machine model.

Recently, deep learning methods have been proven to be effective in text classification. Kim (2014) proposed a convolutional neural network (CNN) architecture that utilized multiple parallel convolutional layers with varying filter window sizes and concatenated the selected important features into a dense softmax layer for sentence classification. Lai et al. (2015) applied a recurrent structure to learn contextual information of each word and employed a max-pooling layer to capture the important features in texts. Another state-of-the-art method is hierarchical attention networks for document classification (Yang et al., 2016). Based on documents' hierarchical structure, it performed attention mechanisms on word-level and sentence-level representations extracted by BiLSTMs.

Such methods have good performance for long texts, especially for documents or paragraphs, but they are inferior when directly applied for short text classification task. Short texts tend to span over a wide range of words, resulting in data sparsity and lack of enough contexts (Chen et al., 2011; Wang et al., 2017).

2.2 Short Text Classification

According to our review, there generally exist two strategies for short text classification.

The first strategy is to leverage an external knowledge base to expand the context of short texts. For example, Feng et al. (2013) calculated the correlation between each short text and domain knowledge for classification. Wang et al. (2014) leveraged a large-scale taxonomy knowledge base to learn the concepts of words and ranked the similarities between short texts and concepts. Wang et al. (2017) associated each short text with its relevant concepts in the knowledge base. They

combined the words and relevant concepts of the short text to generate its embedding. A high-quality knowledge base is vital for their performance, but its construction is time-consuming and labor-intensive, or even worse, it may be unavailable for some domains (Li et al., 2016).

The second strategy is to explore latent topics or clustering features for classification. For example, Chen et al. (2011) derived multi-granularity topics through latent Dirichlet allocation (LDA) as features for traditional classifiers. Ren et al. (2016) used LDA to extract topics and extended existing recursive autoencoder to effectively incorporate topic information. Ma et al. (2015) used Gaussian models to describe the distribution of words embeddings and classified new short texts using the Bayesian rule to get the posterior probability. Revanasiddappa and Harish (2018) developed a fuzzy c-means clustering method and built the match degree between cluster and categories. Such methods can reduce high dimensionality and terms' sparse distribution problems. But their pipeline architecture (i.e., using clustering or topic models to derive clusters or topics, and then integrating them into classifiers as features), might be hard to leverage the mutual dependency of clustering and classification methods.

3 Method

In this paper, we propose a joint model called cluster-gated convolutional neural network (CGCNN), coupling a soft clustering method and a gated CNN for classification. In this section, we mainly introduce the overall architecture of our model, and define the objective function for training.

3.1 Overall Architecture of the Model

Figure 1 presents the CGCNN structure, composing of five major components: (1) a word encoder layer based on BiLSTM to learn word representations in each short text, (2) a clustering layer that calculates words' distributions and performs a linear transformation to get cluster-dependent text representations, (3) a cluster-gated convolutional layer that integrates cluster centers into a gated CNN for further controlling cluster-related feature flows, (4) a max-pooling layer to select most important features and concatenate them as the final text features, and (5) a fully connected layer with softmax function for classification. We update all the parameters in these five components simultaneously, which is introduced in the next subsection.

Word Encoder. Suppose each short text has a maximum of T words, and the t-th word can be

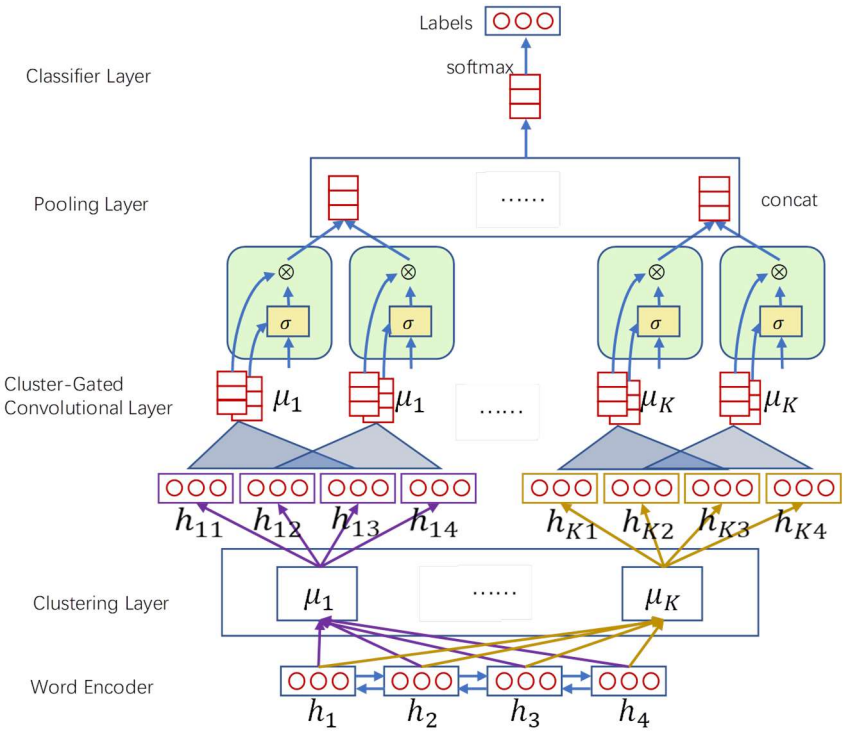


Figure 1: Cluster-gated convolutional neural network.

denoted as w_t, $t \in [1,T]$. We can embed the short text to vectors through an embedding matrix W_e. To capture the local context in text, we employ a BiLSTM to derive the forward representation f_t and backward representation b_t. We concatenate them as word representation, i.e., $h_t = [f_t, b_t]$. Specifically, the input text is represented as a matrix $X = [h_1, h_2, \dots, h_T]$. In some cases with weak sequential text, we will directly use word embedding as the corresponding word representation h_t, i.e., $h_t = W_e w_t$. This will be further discussed in the experiment section.

Clustering Layer. Cluster centers contain semantic closeness of similar words, which is used to selectively control related word flows in the next layer. Here we employ a soft clustering method (Maaten and Hinton, 2008; Xie et al., 2016) to explore words' cluster centers. And then we build a projection function $f_\theta: (h_t, \mu_k) \rightarrow h_{kt}$ to get cluster-dependent text representations, where μ_k refers to the *k*-th cluster center, and h_{kt} refers to the *t*-th representation dependent on the *k*-th cluster center. We set the number of clusters as *K*, i.e., $k \in [1,K]$. The soft clustering method uses the student's *t*-distribution as a kernel to calculate the similarity between word representation h_t and cluster center μ_k, as formula (1).

$$q_{t,k} = \frac{(1+ \| h_t - \mu_k \|^2)^{-1}}{\sum_{k'} (1+ \| h_t - \mu_{k'} \|^2)^{-1}} \quad (1)$$

where $q_{t,k}$ is the probability of *t*-th word belonging to *k*-th cluster. A higher value of $q_{t,k}$ indicates the word is more closed to the cluster.

With the help of the probability, we build a linear function to get the cluster-dependent text representations, as formula (2). It can reduce the role of words unrelated with the cluster, and ensures the sum of all cluster-dependent word representations at position *t* to the corresponding word representation h_t, as formula (3).

$$f_\theta : h_{k,t} = h_t q_{t,k} \quad (2)$$

$$\sum_k h_{k,t} = \sum_k h_t q_{t,k} = h_t \sum_k q_{t,k} = h_t \quad (3)$$

In this way, we can transfer the matrix of a short text to *K* cluster-dependent matrices, as formula (4).

$$X_k = [h_{k,1}, h_{k,2}, ..., h_{k,T}], k \in [1,K] \quad (4)$$

Cluster-Gated Convolutional Layer. The gating mechanism can control information flows in the network, which have been proven effective in LSTM and CNN (Dauphin et al., 2016). With the help of cluster centers, we would further explore related features with clusters in this layer. We employ a convolutional filter $W_k \in R^{D \times n}$ for mapping *n* words into a phrase-level feature, where *D* and *n* refer to the dimention of $h_{k,t}$ and the filter window size respectively. As shifting the filter across the *k*-th cluster-dependent text representation X_k, as formula (5), we can obtain a sequence of new features $C_k = [c_{k,1}, c_{k,2}, \dots, c_{k,L}]$. Here we use no-padding mode, i.e., *L=T-n+1*.

$$c_{k,i} = relu(h_{k,i:i+n} * W_k + b_k) \quad (5)$$

where b_k is the term bias.

Based on gated linear units (GLU) (Dauphin et al., 2016), we use word representations and cluster center to together decide the information passed on, as formulas (6) and (7).

$$g_{k,i} = \sigma(h_{k,i:i+n} * U_k + V_k \mu_k + d_k) \quad (6)$$

$$s_{k,i} = c_{k,i} \otimes g_{k,i} \quad (7)$$

where $U_k \in R^{D \times n}$, $V_k \in R^D$, $d_k \in R$ are learned parameters. σ is the sigmoid function, and $\otimes$ is the element-wise product between vectors. $g_{k,i}$ refers to the cluster-gated value, which is used to control the convolutional feature $c_{k,i}$. And $s_{k,i}$ is the final gated convolutional feature at position *t* for *k*-th cluster-dependent text representation.

Pooling Layer. In this layer, we apply a max-over-time pooling operation over each cluster-gated convolutional features to capture the maximum value as the feature for the corresponding cluster-dependent text representation, as formula (8). And then we concatenate all of them for the next classification layer, as formula (9).

$$s_k = \max\{s_{k,i}, i \in [1,T]\}, k \in [1,K] \quad (8)$$

$$s = s_1 \oplus s_2 \oplus \dots \oplus s_K \quad (9)$$

Classifier Layer. For each short text instance, we generate the high-level representations of the combination of multiple clusters' related information. To make full use of them, we use a fully connection with softmax function for prediction. The probability assigning a category label to this instance, can be calculated as formula (10).

$$p(\hat{y} = j) = soft\max(Ws + b) \quad (10)$$

where j is the category label. To avoid over-fitting, we can also employ dropout in this layer.

3.2 Training

The entire CGCNN model integrates a clustering method into the gated-CNN for classification, which can be updated simultaneously in one framework. Hence, we combine their loss effects into one objective function as formula (11).

$$L = L_{CLF} + \lambda L_{CLU} \quad (11)$$

where L_{CLF} is the cross-entropy loss of the classifier, and L_{CLU} is the clustering loss with Kullback-Leibler divergence (KL divergence) minimization. $\lambda > 0$ is a tradeoff parameter controlling the degree of clustering loss. The classifier loss can be defined as formula (12).

$$L_{CLF} = -\sum_i \sum_j 1\{y_i = j\} \log P(\hat{y}_i = j) \quad (12)$$

where i is the i-th sample instance, y_i is the ground truth label, and $1\{*\}$ is the indicator function.

For the clustering loss, we use KL divergence between the distribution of soft labels $q_{t,k}$ and the auxiliary distribution $p_{t,k}$ as (Maaten and Hinton, 2008; Xie et al., 2016), as formula (13).

$$L_{CLU} = \sum_i \sum_t \sum_k p_{t,k} \log \frac{p_{t,k}}{q_{t,k}} \quad (13)$$

where $p_{t,k}$ is the target distribution, as formula (14).

$$p_{t,k} = \frac{{q_{t,k}}^2 / \sum_{t'} q_{t',k}}{\sum_{k'} ({q_{t,k'}}^2 / \sum_{t'} q_{t',k'})} \quad (14)$$

As (Xie et al., 2016), this target distribution is computed by first raising the second power of $q_{t,k}$ to its corresponding soft cluster frequencies $\sum_{t'} q_{t',k}$ and then performing normalization to prevent large clusters from distorting the hidden feature space. It can not only improve cluster purity, but also emphasize the data points assigned to clusters with high confidence.

4 Experiments

4.1 Datasets and Preprocessing

To illustrate the effectiveness of our model, we conduct experiments on five public datasets: AG News, Sogou News, Amazon Reviews, Yahoo! Answers, and Search Snippets. The first three datasets are adopted from (Zhang et al., 2015b). The last two datasets are from the Yahoo! Webscope program and (Phan et al., 2008) respectively. For each dataset, we use 80% of the data for training, 10% for validation, and the remaining 10% for test. To construct short texts, we only use titles or some partial information of the datasets.

AG News and Sogou News. These two original datasets include 127,600 samples from 4 categories and 510,000 samples from 5 categories respectively. Sogou News is a dataset in Chinese, and Zhang et al. (2015b) combined pypinyin package and a Chinese segmentation tool to produce Pinyin – Roman spelling in Chinese. For both of them, each sample contains both title and content of news. To test for short texts, we remove contents and only use the titles in our experiment.

Amazon Reviews. The full dataset contains 3.65 million samples from one-to-five rating labels. In order to test for short texts, we remove the review contents and only use the review titles in our experiment.

Yahoo! Answers. This corpus includes 4,483,032 question titles, question contexts and their answers. We use 10 largest classes to construct a topic classification task. We randomly choose 50,000 samples for each class. Here we only use the question titles for classification.

Search Snippets. This dataset, released by Google search engine, includes 12,340 samples with predefined 8 categories by (Phan et al., 2008).

Note that we filter out punctuation and use Natural Language Toolkit (NLTK) for stemming. We do not remove stopwords since some of them may carry classification information, especially for users' reviews. The details of each dataset are listed in Table 1.

Datasets	Size	Classes	Avg. Len
AG News	127,600	4	7.0
Sogou News	510,000	5	15.4
Amazon Review	3,650,000	5	4.6
Yahoo! Answers	500,000	10	11.2
Search Snippets	12,340	8	17.9

Table 1: A summary of datasets.

4.2 Implementation Detail

The model hyper-parameters are tuned based on the AG News dataset. We also conduct experiments with the model directly using word embeddings instead of BiLSTM, representing as CGCNN*. We firstly set the dimension of word embeddings to 300, and pre-train word embeddings on each dataset with word2vec. The dimensions of all hidden vectors are set to 200. For the clustering method, we set the number of clusters to the number of ground-truth categories, and randomly initialize cluster center vectors. We set λ=0.5 and λ=0.6 for CGCNN* and CGCNN respectively to control the effects of clustering method. To avoid model over-fitting, we use dropout with rate of 0.2. We train the parameters by using Adam method with a learning rate of 0.001, and set the batch size to 64. The filter sizes of all convolution layers are set to 3 in these two methods.

4.3 Baselines and Experimental Settings

In this paper, we choose the following baseline algorithms for comparison:

CNN (Kim, 2014). It builds a multi-channel convolutional architecture with varying filter window sizes, and concatenates the important features extracted by a max-over-time pooling operation.

CNNM. To further illustrate the effectiveness of our model, we develop the multi-channel convolutional architecture (Kim, 2014) with multiple fixed size filters. As our model hyperparameters, the number of filters is equal to the number of ground-truth categories, and all their sizes are set to 3.

RCNN (Lai et al., 2015). It develops a recurrent convolutional structure. It employs a bi-directional recurrent structure to capture word context embeddings and uses a max-pooling layer to select the important features.

CNN-LSTM (Zhou et al., 2015). This method uses a multi-channel convolutional layer to extract higher-level phrase features, and employs a BiLSTM to capture their sequences for classification.

AttBiLSTM (Lin et al., 2017). It uses a BiLSTM to explore the sequences of texts, and develops a self-attention mechanism to get sentence-level representations.

For the multiple-channel convolutional architecture of CNN and CNN-LSTM, the filter sizes are 3, 4 and 5, as Kim (2014)'s default settings. For the hidden vectors of BiLSTM in these methods, we also set their dimensions to 200.

4.4 Results

We use accuracy as the evaluation metric, and Table 2 reports the different algorithms' performance on the five real-world datasets. We highlight the highest value in each column. As we can see, either CGCNN or CGCNN* has the best performance on the datasets. The CNN-LSTM outperforms the other baseline methods on AG News, Amazon Review and Yahoo! Answers datasets, while CNN and CNNM have the best performance on Sogou News and Search Snippets respectively. As compared with CNN-LSTM, CGCNN has about 1.5% performance improvements on Amazon Review and Yahoo! Answers, and 0.49% on AG News dataset. CGCNN* can achieve 0.6%~0.8% performance improvements over the second best baseline method on Sogou News and Search Snippets datasets. The AttBiLSTM method has poor performance. We suspect that lack of enough context might cause the failure of the self-attention mechanism.

	AG News	Sogou News	Amazon Review	Yahoo! Answers	Search Snippets
CNN	87.62%	90.47%	46.71%	61.64%	93.60%
CNNM	87.89%	90.17%	46.65%	61.59%	93.84%
CNN-LSTM	88.12%	89.67%	47.80%	62.61%	93.11%
RCNN	87.69%	88.43%	46.95%	61.90%	93.68%
AttBiLSTM	87.63%	87.63%	46.96%	61.92%	91.09%
CGCNN*	88.24%	**91.02%**	47.17%	63.01%	**94.57%**
CGCNN	**88.55%**	90.95%	**48.65%**	**63.55%**	92.30%

Note: CGCNN* represents that our model directly inputs word embeddings into the clustering layer.

Table 2: Accuracy comparison on different datasets.

The CNNM method using category number of convolutional filters, has similar performance with the CNN method using three convolutional filters, which illustrates increasing number of convolutional filters might have no active impact on performance. Differently, our CGCNN* method using category number of clusters for gated CNN, achieves better accuracies than both of them. It shows that our proposed architecture, integrating a clustering-gated mechanism into CNN, can significantly improve the performance in short text classification.

The CNN-LSTM method uses CNN and BiLSTM to capture phrase features and their sequences, outperforms CNN and CNNM on AG News, Amazon Reviews and Yahoo! Answers datasets, while it has poorer performance on Sogou News and Search Snippets datasets. Such cases also exist in the comparison between CGCNN and CGCNN* methods. We analyze the datasets, and suspect that weak sequential relationship in texts may result in the decreasing performances on Sogou News and Search Snippets. The original Sogou News was transferred from Chinese characters to Pinyin format (Zhang et al., 2015b). It might cause a homophone problem. For example, word "与(and)" and word "雨(rain)" have the same pronunciation but different meanings in Chinese. It breaks sequential patterns in texts. For Search Snippets, each sample is consisted of multiple keywords, and there are no obvious sequences among them. For example, a sample likes "… calorie count calories item ...", containing weak sequential semantics.

4.5 Clustering Analysis

To further study the impact of clustering method, we conduct additional experiments on AG News dataset by varying the tradeoff parameter λ and the cluster number K, and assess the sensitivity of our model.

Figure 2 reports the change of performance with increasing values of tradeoff parameter λ from 0.1 to 0.9 while keeping the cluster number K constant (as the category number). We can observe that CGCNN* and CGCNN reach the best performances when $\lambda = 0.5$ and $\lambda = 0.6$ respectively. When tradeoff parameter λ varies from 0.1 to the values of their best performances, their performances generally show an increasing trend, which implies the clustering effect can benefit for understanding short texts. When tradeoff parameter λ increases from the optimal values to 0.9, the performances of these two methods generally have a slight decrease, which shows excessive clustering might have a bad influence on short text classification.

Figure 3 reports the results of adjusting the number of clusters (K) in CGCNN* and CGCNN when we set λ to the optimal values (i.e., $\lambda = 0.5$ and $\lambda = 0.6$ respectively). For CGCNN method, we can observe that it reaches the best performance when the cluster number equals to the category number (i.e., $K = 4$). No matter the cluster number increases or decreases, its performance would have a decrease tendency. While the CGCNN* method's performance generally show an increasing trend, which relatively stabilizes when K reaches category number (although its performance has a slight decrease at $K = 5$). That is the reason that we set the cluster number to the category number.

4.6 Case Study

In this section, we take several concrete samples from AG News dataset to illustrate how the

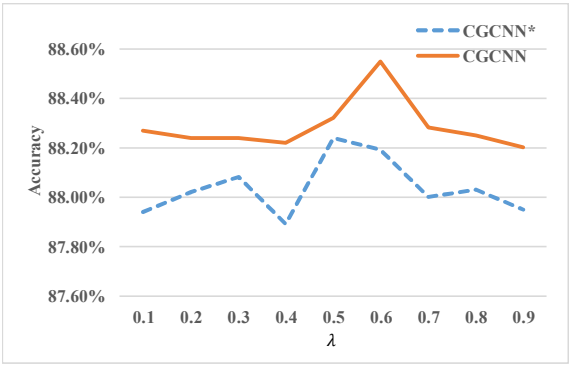


Figure 2: Performance with tradeoff parameter λ on AG News.

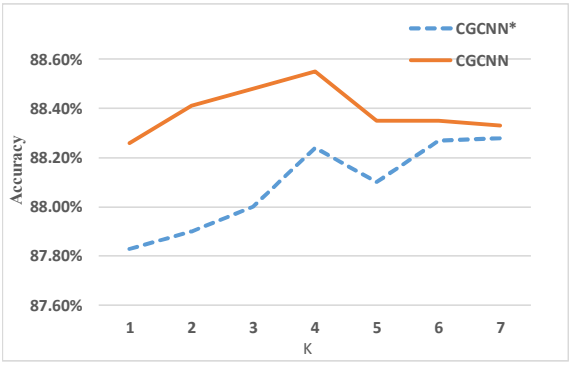
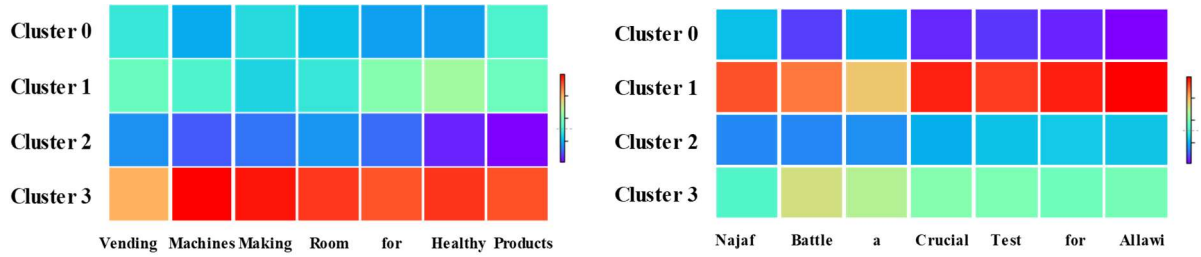
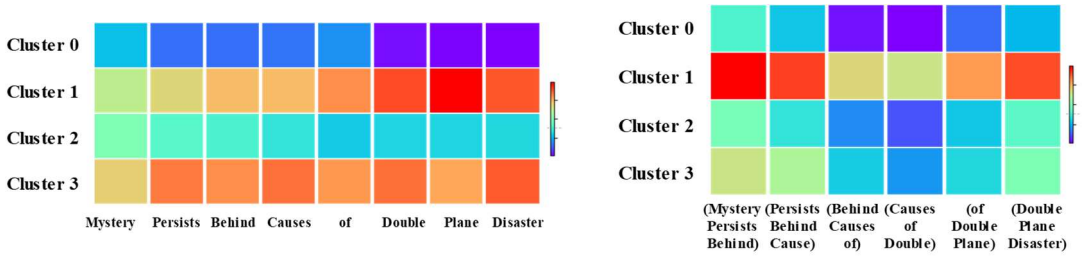


Figure 3: Performance with cluster number K on AG News.



(a) An instance with category "Sci/Tech" (b) An instance with category "World"

Figure 4: The similarities between words and clusters in a short text.



(a) The similarities between words and clusters in an instance with category "World"

(b) The outputs of the corresponding cluster-gated convolutional features

Figure 5: The role of the cluster-gated layer.

proposed method works. Here we use CGCNN method, because of strong sequential relationship in this dataset.

In the clustering layer, we leverage a soft clustering method to explore words' cluster centers, and build a linear function to project the representation of a short text to K cluster-dependent representations. Here we calculate the similarity between words and cluster centers, and normalize the values of each word belonging to clusters. Figure 4 (a) and (b) show two different instances from categories "Sci/Tech" and "World" respectively. We can observe the linear projection can strengthen the words' representations dependent on some cluster, and weaken them on others. These two figures have different distributions on the same word "for", which is due to different contexts explored by BiLSTM.

There might exist some instances closely related with two or more clusters, as figure 5 (a). To further control the information flows, we leverage cluster centers and phrase-level features for the gated mechanism. We use Xue and Li (2018)'s method to visualize the gated mechanism: summing the representation of each phrase-level feature and normalizing them according to clusters. Figure 5 (a) shows the similarities between words and clusters in an instance with category "World", while figure 5 (b) shows the corresponding cluster-gated convolutional features. We can observe that cluster-gated layer can further strengthen the corresponding cluster-dependent representation, and weaken others.

5 Conclusion

In this paper, we propose a joint model that couples clustering and classification methods. It employs a BiLSTM to learn word representations for local contexts in short texts. We take a soft clustering method to calculate the probability of each word assigning to each cluster, which can derive the semantic relation of word representations and the global corpus. We also perform a linear transformation to explore cluster-dependent text representations. Moreover, we develop a cluster-gated CNN by integrating cluster centers into GLU, which can select cluster-related features for classification. Experiments on five real-world datasets show that our model does better than the state-of-the-art methods for short text classification task.

In the future work, we will further analyze the mutual effects of document-level clustering and classification methods for long text, and attempt to develop more effective joint model for text classification. Moreover, we will study some other

mechanisms (e.g., highway units, attention mechanism) to further improve the performance.

6 Acknowledge

We thank Junjie Li, Jianwei Guo, Yiqiang Shi, and the anonymous reviewers for the constructive suggestions on various aspects of this work.

References

Chen, M., X. Jin and D. Shen. 2011. Short text classification improved by learning multi-granularity topics. Proceedings of the 22th International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, AAAI Press, pages 1776-1781.

Dauphin, Y. N., A. Fan, M. Auli and D. Grangier. 2016. Language modeling with gated convolutional networks. Proceedings of the 34th International Conference on Machine Learning. 70**,** pages 933-941.

Feng, X., Y. Shen, C. Liu, W. Liang and S. Zhang. 2013. Chinese Short Text Classification Based on Domain Knowledge. International Joint Conference on Natural Language Processing, Nagoya, Japan, pages 859–863.

Gehring, J., M. Auli, D. Grangier, D. Yarats and Y. Dauphin. 2017. Convolutional Sequence to Sequence Learning. Proceedings of the 34th International Conference on Machine Learning (ICML), pages 1243-1252.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) University of Waterloo, pages 1746-1751.

Lai, S., L. Xu, K. Liu and J. Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. Proceedings of the 29-th AAAI Conference on Artificial Intelligence, pages 2267-2273.

Lazaridou, A., I. Titov and C. Sporleder. 2013. A Bayesian Model for Joint Unsupervised Induction of Sentiment, Aspect and Discourse Representations. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria, Association for Computational Linguistics, pages 1630-1639.

Li, C., H. Wang, Z. Zhang, A. Sun and Z. Ma. 2016. Topic Modeling for Short Texts with Auxiliary Word Embeddings. Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. Pisa, Italy, ACM**,** pages 165-174.

Lin, Z., M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou and Y. Bengio. 2017. A Structured Self-Attentive Sentence Embedding. International Conference on Learning Representations 2017 (ICLR), pages.

Luo, G., X. Huang, C. Y. Lin and Z. Nie. 2015. Joint Named Entity Recognition and Disambiguation. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, ACL, pages 879-888.

Ma, C., W. Xu, P. Li and Y. Yan. 2015. Distributional Representations of Words for Short Text Classification. Proceeding of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics. Denver, Colorado**,** pages 33-38.

Maaten, L. v. d. and G. Hinton. 2008. Visualizing data using t-SNE. Journal of machine learning research. 9(Nov): 2579-2605.

Phan, X.-H., L.-M. Nguyen and S. Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. Proceedings of the 17th International Conference on World Wide Web. Beijing, China, ACM**,** pages 91-100.

Post, M. and S. Bergsma. 2013. Explicit and Implicit Syntactic Features for Text Classification. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria,, Association for Computational Linguistics, pages 866-872.

Ren, Y., R. Wang and D. Ji. 2016. A topic-enhanced word embedding for Twitter sentiment classification. Information Sciences. 369: 188-198.

Revanasiddappa, M. and B. Harish. 2018. A New Feature Selection Method based on Intuitionistic Fuzzy Entropy to Categorize Text Documents. International Journal of Interactive Multimedia & Artificial Intelligence. 5(3): 106-117.

Schmitt, M., S. Steinheber, K. Schreiber and B. Roth. 2018. Joint Aspect and Polarity Classification for Aspect-based Sentiment Analysis with End-to-End Neural Networks. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, pages 1109–1114.

Shao, Y., C. Hardmeier, J. Tiedemann and J. Nivre. 2017. Character-based Joint Segmentation and POS Tagging for Chinese using Bidirectional RNN-CRF. Proceedings of the The 8th International Joint Conference on Natural Language Processing, Taipei, Taiwan, Asian Federation of Natural Language Processing, pages 173-183.

Wang, F., Z. Wang, Z. Li and J.-R. Wen. 2014. Concept-based Short Text Classification and Ranking. Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, Shanghai, China, pages 1069-1078.

Wang, J., Z. Wang, D. Zhang and J. Yan. 2017. Combining knowledge with deep convolutional neural networks for short text classification. Proceedings of the 26th International Joint Conference on Artificial Intelligence. Melbourne, Australia, AAAI Press, pages 2915-2921.

Xie, J., R. Girshick and A. Farhadi. 2016. Unsupervised deep embedding for clustering analysis. Proceedings of the 33rd International Conference on Machine Learning, pages 478-487.

Xue, W. and T. Li. 2018. Aspect Based Sentiment Analysis with Gated Convolutional Networks. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics Association for Computational Linguistics, pages 2514-2523.

Yang, Z., D. Yang, C. Dyer, X. He, A. Smola and E. Hovy. 2016. Hierarchical attention networks for document classification. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, pages 1480-1489.

Zhang, D., H. Xu, Z. Su and Y. Xu. 2015a. Chinese comments sentiment classification based on word2vec and SVMperf. Expert Systems with Applications. 42(4): 1857-1863.

Zhang, X., J. Zhao and Y. LeCun. 2015b. Character-level convolutional networks for text classification. Advances in neural information processing systems, pages 649-657.

Zhou, C., C. Sun, Z. Liu and F. C. M. Lau. 2015. A C-LSTM Neural Network for Text Classification. Computing Research Repository. arXiv:1511.08630.

Coherence-based Modeling of Clinical Concepts Inferred from Heterogeneous Clinical Notes for ICU Patient Risk Stratification

Tushaar Gangavarapu* tushaargvsg45@gmail.com
Gokul S Krishnan gsk1692@gmail.com
Sowmya Kamath S sowmyakamath@nitk.edu.in
Healthcare Analytics and Language Engineering (HALE) Lab
Department of Information Technology
National Institute of Technology Karnataka, Surathkal, Mangaluru, India

Abstract

In hospitals, critical care patients are often susceptible to various complications that adversely affect their morbidity and mortality. Digitized patient data from Electronic Health Records (EHRs) can be utilized to facilitate risk stratification accurately and provide prioritized care. Existing clinical decision support systems are heavily reliant on the structured nature of the EHRs. However, the valuable patient-specific data contained in unstructured clinical notes are often manually transcribed into EHRs. The prolific use of extensive medical jargon, heterogeneity, sparsity, rawness, inconsistent abbreviations, and complex structure of the clinical notes poses significant challenges, and also results in a loss of information during the manual conversion process. In this work, we employ two coherence-based topic modeling approaches to model the free-text in the unstructured clinical nursing notes and capture its semantic textual features with the emphasis on human interpretability. Furthermore, we present *FarSight*, a long-term aggregation mechanism intended to detect the onset of disease with the earliest recorded symptoms and infections. We utilize the predictive capabilities of deep neural models for the clinical task of risk stratification through ICD-9 code group prediction. Our experimental validation on MIMIC-III (v1.4) database underlined the efficacy of *FarSight* with coherence-based topic modeling, in extracting discriminative clinical features from the unstructured nursing notes. The proposed approach achieved a superior predictive performance when benchmarked against the structured EHR data based state-of-the-art model, with an improvement of 11.50% in AUPRC and 1.16% in AUROC.

*Corresponding author.

1 Introduction

Until recently, the healthcare industry had an inclination towards conservative approaches for the treatment and diagnosis of patients, resulting in less patient-centric and imprecise assessments (Mathew and Pillai, 2015). Intensive Care Units (ICUs) utilize the most advanced medical resources to treat and monitor critically ill patients. However, such advanced medical interventions in ICUs often make patients vulnerable to several complications (To and Napolitano, 2012). Various infections, including barotrauma, short- and long-term intubation, catheter-associated urinary tract infection, weaning errors, ventilator-associated pneumonia, gastrointestinal tract bleeding, and infections from unrecognized drug interactions, are associated with invasive ICU devices (Wollschlager and Conrad, 1988). The lack of accurate knowledge of the etiology of such complications leads to the inability to accurately stratify risk, due to which, in most cases, adequate care is provided to patients only after the development of a complication (Huddar et al., 2016). With the advent of digitization, advancement in technology, need for evidence-based medicine, increased population, and rising rates of chronic diseases, the utilization of ever-increasing heterogeneous medical data to improve the quality of life has become imperative. Specifically, ICUs are data-rich environments where several parameters of patients are monitored continuously. Such data can be vital to improve the existing Clinical Decision Support Systems (CDSSs), develop new treatments, and predict prominent clinical events and outcomes. Furthermore, such CDSSs could promote evidence-based and patient-centric treatments, resulting in reduced hospital mortality and morbidity rates, and improved risk assessment.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 1012–1022
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

Pat is 83 yo F w/PMHx for CLL and hypotens, who was admited for an elective total hip arthroplasty for persistent hip pain. NGT to low cont suct. Family here to visit.

Pat initially sustained a right hip fracture after a fall in [**2137**], and had an ORIF performed at the time. Gave med for pain. Has had right hip pain ever since, and also has AVN of the right femoral head.

She came in today for elective tot hip repl. In the OR today, patient had an estimated 1600cc EBL, and received 6u pRBC. I/Os were 7200cc in (3.7L LR, 1.5L pRBCs).

Figure 1: Sample de-identified nursing note from critical care. Observe the absence of grammatical structure, informal word usage, and extensive medical jargon.

Structured medical data in the form of Electronic Health Records (EHRs) contain numerical assessments (e.g., lab results) and are amenable to standard statistical analysis (Huddar et al., 2016). However, unstructured clinical text and images also contain valuable information concerning the state of a patient. In particular, clinical nursing notes maintain objective and subjective assessments of a patient's condition. Such raw notes contain the intuitions and observations of nurses and caregivers who regularly monitor the patient. This valuable patient-specific information present in the clinical nursing notes has the potential to uncover hidden clues about the mental state (e.g., family support and mental fitness) and the health of a patient (Jo et al., 2015). Such information is not found in EHRs or elsewhere (Dubois et al., 2017). However, these notes are informally written, and modeling such notes is challenging due to their high-dimensionality, rawness, sparsity, complex linguistic and temporal nature, inconsistent abbreviations, and occurrence of rich medical jargon (a sample note is shown in Figure 1).

The voluminosity of nursing notes can be observed from the heavy-tailed distribution of the MIMIC-III nursing notes across various patients (see Figure 2), with an average of 176.49 nursing notes per patient. The presentation, analysis, and interpretation of the data present in such notes in a medically appropriate and usable format determine the competence of the underlying CDSS (Wang et al., 2018). Furthermore, there is often a need to assign multiple labels to a patient entry, owing to the diverse and manifold nature of the disease symptoms of the patients (Baumel et al., 2018). Risk stratification as ICD-9[1] code group prediction can help in predicting disease onset and its severity, thus facilitating preventive and prioritized care, and reduction of hospital mortality and morbidity rates.

With the availability of large de-identified healthcare databases such as MIMIC-III[2] (Johnson et al., 2016), modeling patient data using machine and deep learning to predict prominent clinical events and outcomes has sparked widespread interest. Early works (Tu and Guerriere, 1993; Doig et al., 1993; Grigsby et al., 1994; Clermont et al., 2001; Hanson and Marshall, 2001) have reported on the superior performance of machine learning models in forecasting the length-of-stay and mortality, for ICU patients. More recently, Pirracchio (2016) used an ensemble of several machine learning models that offered improved performance in ICU mortality prediction over various

[1]International Classification of Diseases, ninth revision.
[2]Medical Information Mart for Intensive Care.

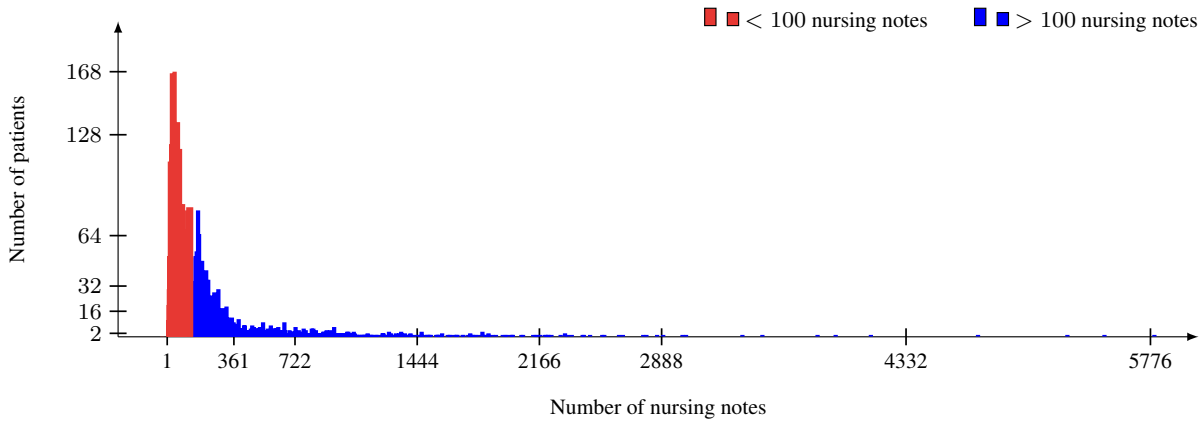


Figure 2: Distribution of the nursing notes across various MIMIC-III subjects.

severity scoring systems. Feldman et al. (2016) mined the clinical nursing, radiology, physician, and ECG narratives to study the linguistic, structural, and topical differences among them. The authors only provided a foundation for mining clinical notes effectively, and in our work, we extend their efforts by effectively modeling the underlying patient representations of the nursing text through effective topic modeling and deep neural learning. Johnson et al. (2017) extracted a set of features from the MIMIC-III database for ICU mortality prediction and compared several state-of-the-art models against gradient boosting and logistic regression. The authors stressed the need for improvement in the way of reporting performance to ensure a fairer comparison. Most of these models utilize machine learning models built on structured EHR data for the prediction of clinical tasks.

Recent works show promising results in modeling patient data using deep learning approaches. Harutyunyan et al. (2017) benchmarked their performance on four clinical prediction tasks on the MIMIC-III database using multitask recurrent neural networks. Zalewski et al. (2017) presented a viable framework to combine several modalities of a patient's health states for risk stratification. Their approach was built on the hierarchical Dirichlet method, aimed at tackling the sparsity and high-dimensionality of the nursing notes extracted from the MIMIC-II database. However, the authors used a logistic regression model to predict the mortality rate of the patients and did not evaluate their performance with the recent works and deep neural architectures. Purushotham et al. (2018) reported a suite of five clinical prediction tasks, including the length-of-stay, mortality, and ICD-9 code group prediction on the MIMIC-III database using deep learning models and benchmarked their performance against the existing state-of-the-art methods and severity scoring systems. However, mining and modeling the valuable patient-specific information in unstructured clinical nursing notes for the development of CDSSs remains mostly uncommon.

In this paper, we discuss an approach to model the rich patient-specific information in the unstructured clinical nursing notes, to aid in the risk stratification as an ICD-9 code group prediction task. ICD-9 codes are a taxonomy of diagnostic codes used for cost-effectiveness analysis, epidemiology studies, and designing healthcare policies. Accurate ICD-9 code group prediction not only promotes better ICD-9 code determination, but also facilitates more reliable risk stratification by reporting on the severity, symptoms, and the use of resources across code groups, thus aiding disease-specific staging systems. In our work, two coherence-based topic modeling approaches, Coherence-based Latent Dirichlet Allocation (C-LDA) and Coherence-based Nonnegative Matrix Factorization (C-NMF) are employed to capture the semantic relationships between the textual features of the clinical notes and derive optimal data representations with a higher guarantee on human interpretability. We employ *FarSight* to aggregate the documented patient data in a way intended to detect the onset of the disease with the earliest recorded symptoms. Furthermore, we benchmark the performance of our proposed topic models using two neural architectures, including Multi-Layer Perceptron (MLP) and Attention-based Long Short Term Memory (A-LSTM). Additionally, we perform a sensitivity analysis to assess the statistical significance of the obtained results.

The remainder of this paper is structured as follows: Section 2 describes the MIMIC-III database, the preprocessing steps, and the topic modeling approaches employed to obtain the optimal data representations from the raw clinical nursing notes. The deep neural architectures employed in the clinical task of ICD-9 code group prediction along with the discussion of the experimental results of our benchmarking are presented in Section 3. Finally, Section 4 summarizes this paper with highlights on future research possibilities.

2 Materials and Methods

In this section, we discuss in detail, the Natural Language Processing (NLP) pipeline designed to facilitate multi-label ICD-9 code group prediction, and the same is depicted in Figure 3.

2.1 Dataset and Cohort Selection

MIMIC-III (v1.4) is a publicly available large healthcare database with comprehensive medical data of over $40,000$ ICU patients. The healthcare database contains $223,556$ nursing notes extracted from $2,083,180$ note events (*noteevents* table), corresponding to $7,704$ distinct patients (*diagnoses_icd* table). Two selection criteria were employed in the cohort selection. Firstly, only

those records corresponding to the patients older than 15 (adults) were retained using the patient's age at the time of admission to the ICU (extracted from *admissions* and *patients* tables). Secondly, only the first admission of a patient to the hospital was considered. Both these steps were followed in accordance with the existing literature (Johnson et al., 2017; Purushotham et al., 2018). The resultant dataset comprises nursing notes of $7,638$ patients with a median age of 66 years (Quartile $Q_1 - Q_3$: $52 - 78$ years).

2.2 Data Cleaning

The data extracted from the MIMIC-III database contained erroneous patient entries due to several factors, including missing values, duplicate or incorrect records, outliers, and noise. The erroneous entries were filtered out using the *iserror* attribute of the *noteevents* table. Then, duplicate patient records were identified and deduplicated. The resultant dataset comprised of nursing notes corresponding to $6,532$ patients, and the data in these records were aggregated using the proposed *FarSight* technique.

2.3 FarSight: Long-Term Aggregation

It is crucial to detect the onset of the disease with the earliest detected symptoms, to provide preventive care and reduce the mortality and morbidity of complications. We propose *FarSight*, which is designed to aggregate the patient data using a future lookup on all the detected diseases in the later medical records concerning that patient. Let $\mathcal{P}$ be the set of all patients, and let a patient p have a sequence of N clinical notes, $\mathcal{S}^{(p)} = \{(\eta_i^{(p)}, \mathcal{I}_i^{(p)})\}_{i=1}^{N}$, with each clinical note $\eta_i^{(p)}$ mapped to an ICD-9 code $\mathcal{I}_i^{(p)}$ indexed in the order from the oldest to the most recent. Now, *FarSight* aggregates the ICD-9 codes across the nursing notes of a patient using a future lookup, resulting in $\mathcal{S}^{(p)} = \{(\eta_i^{(p)}, \mathcal{I}^{(p)})\}_{i=1}^{N}$, where $\mathcal{I}^{(p)} = \{\mathcal{I}_i^{(p)}\}_{i=1}^{N}$. Ultimately, we aim at learning a function $\mathcal{F}$ to estimate the probability of classifying a given nursing note $\eta_j^{(p)}$ into a set of diagnostic code groups: $\mathcal{F}(S^{(p)}) \approx Pr(\mathcal{I}^{(p)} \mid \eta_j^{p})$. Instead of aggregating several patient records, *FarSight* only aggregates the ICD-9 codes across a particular patient's nursing notes to facilitate risk stratification at the initial stages of the disease with the earliest recorded symptoms and infections.

2.4 Data Preprocessing

Data (text) normalization is performed to facilitate the transformation of inconsistent and informally written medical text into a consistent canonical form. Preprocessing includes tokenization, stopword removal, and stemming/lemmatization. Tokenization splits the nursing text into words (tokens). Using the NLTK English stopword corpus, we removed the stopwords from the generated set of tokens. Next, references to images (e.g., *PET_Scan.jpg*) were removed, and character case folding was performed. Word length based token removal was not performed to retain medical abbreviations such as *CT*, *MRI*, *DEXA*, and *PET*. Lastly, stemming was employed to facilitate suffix stripping, followed by lemmatization to convert the stripped tokens into their base forms. The tokens appearing in less than ten clinical notes were eliminated to mitigate overfitting and lower the computational complexity of training.

2.5 Topic Modeling of Clinical Notes

Let the set of all nursing notes be $\mathbb{S} = \{\mathcal{S}^{(p)}\}_{p=1}^{\mathcal{P}}$. Each nursing note η_j constitutes a variable length of words from a large vocabulary $\mathbb{V}$, making $\mathbb{S}$ very complex. Thus, a transformation (T) of the unstructured clinical text to a machine-processable form ($T : \mathbb{S} \rightarrow \mathbb{R}^k$ ($k \ll |\mathbb{V}|$)) is vital to the efficacy and performance of the underlying deep neural architectures.

Topic modeling aims at finding a set of topics

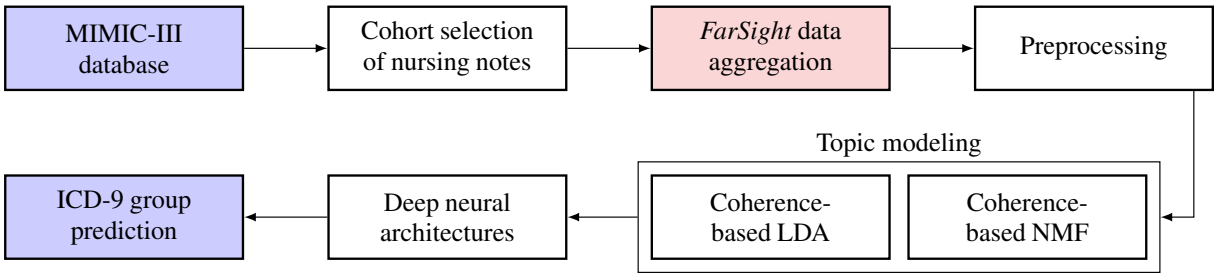


Figure 3: NLP pipeline used in the prediction of the ICD-9 code group.

from a set of clinical notes that best represents the corpus. Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a cluster analysis approach based on the three-layer Bayesian framework including documents, topics, and tokens. LDA draws a mixture of topics from the Dirichlet distribution and facilitates a flat and soft probabilistic clustering of tokens into topics and documents into topics. LDA posits that each term and clinical note belong to a set of clinical topics with a certain probability. Nonnegative Matrix Factorization (NMF) (Lee and Seung, 2000) is a matrix factorization approach that decomposes multivariate data into topics. In NMF, each topic is a nonnegative linear combination of the tokens in the vocabulary. NMF iteratively decomposes the data matrix ($N \times |\mathbb{V}|$) into two lower rank matrices with $\mathcal{T}$ topics ($N \times \mathcal{T}$ and $\mathcal{T} \times |\mathbb{V}|$). These topic models capture the context of occurrence and co-occurrence, which is essential for accurate predictability of the underlying deep neural models.

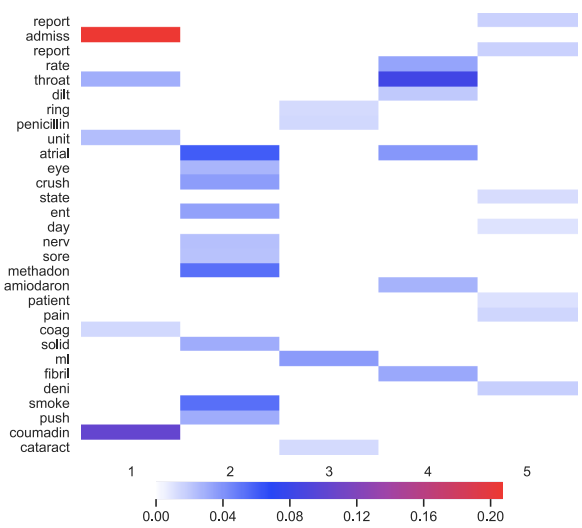
Determining the optimal number of LDA or NMF clusters is a challenging task. To address this issue, we utilize the Topic Coherence (TC) or semantic coherence (Röder et al., 2015) between the topics to derive the optimal number of clusters. Furthermore, when topics are learned from a multinomial distribution over words from noisy and sparse text data, they are less coherent and hard to interpret. TC evaluates topic models with a greater guarantee of human interpretability. This study adopts LDA and NMF with TC (C-LDA and C-NMF) as TC accounts for the semantic similarity between the higher scoring tokens and facilitates the generation of human-understandable topics. We employ the C_v variant of coherence measurement with a Normalized Pointwise Mutual Information (NPMI) score (Bouma, 2009) as the confirmation measure, due to its high correlation with the available human-judged data (Röder et al., 2015). Let $\mathcal{T} = \{t_1, t_2, \ldots, t_k\}$ be a topic generated from a topic model which is represented using its top-k most probable tokens (t_is). Note that higher values of the average pairwise similarity among the tokens in $\mathcal{T}$ imply greater coherence of the topic. For a predetermined similarity measure $S(t_i, t_j)$ (here NPMI), the coherence score is computed as:

$$\text{Coherence}_S(\mathcal{T}) = \frac{\sum_{1 \le i \le k-1} \sum_{i+1 \le j \le k} S(t_i, t_j)}{\binom{k}{2}} \quad (1)$$

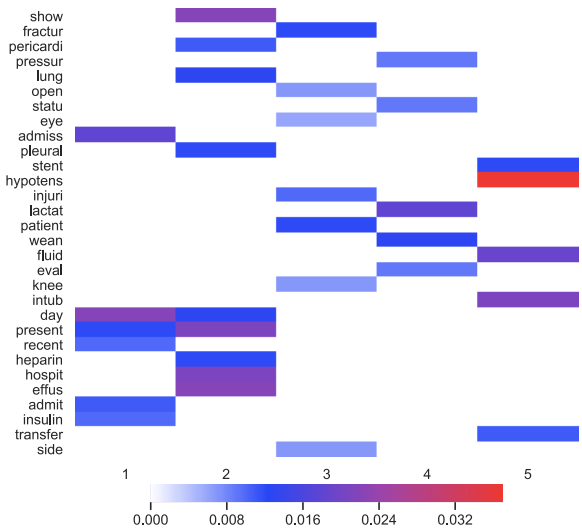
where $t_i, t_j \in \mathcal{T}$. The coherence score comes from external data, i.e., the data not used during training (we employed the full set of English Wikipedia articles), and is intended to regularize the topic models. The NPMI similarity score is an extension of the pointwise mutual information score, and is used in finding associations and collocations between the words (Aletras and Stevenson, 2013). The NPMI score is computed as:

$$\text{NPMI}(t_i, t_j) = \frac{\text{PMI}(t_i, t_j)}{-\log_2(\Pr(t_i, t_j))} \quad (2)$$

$$\text{PMI}(t_i, t_j) = \log_2 \left(\frac{\Pr(t_i, t_j)}{\Pr(t_i)\Pr(t_j)} \right) \quad (3)$$



(a) Coherence-based LDA.



(b) Coherence-based NMF.

Figure 4: Correlations between top terms' membership in top five topic modeling clusters.

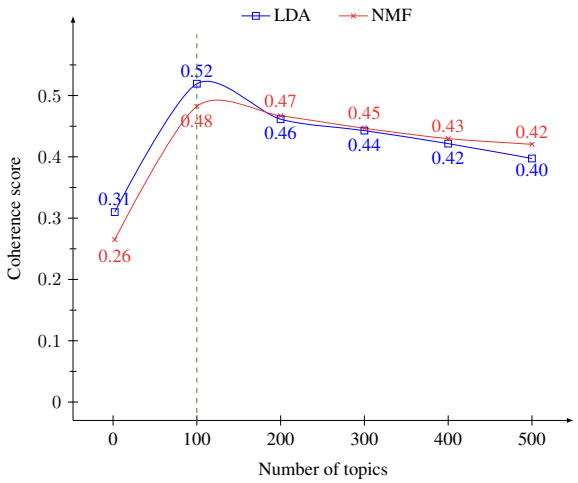


Figure 5: Coherence score comparison to determine the optimal number of topics.

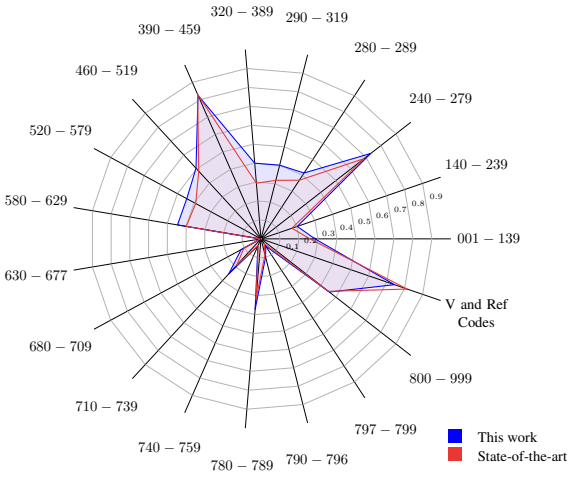


Figure 6: Comparison of ICD-9 code group statistics with the state-of-the-art model (Purushotham et al., 2018).

The individual confirmation measures obtained for all topics ($\mathcal{T}_i$s) are averaged to obtain the final coherence score.

The number of topics for both LDA and NMF models was determined to be 100, by computing the coherence score of several topic models obtained by varying the number of topics. The LDA and NMF matrices were built on a bag-of-words representation of the clinical notes. For the ease of interpretation, a heat map presenting the correlations between top terms' membership in top five C-LDA clusters is presented in Figure 4a, and top five C-NMF clusters is depicted in Figure 4b. From Figure 4, it can be observed that both the C-LDA and C-NMF models effectively capture specific clinical terms, including *penicillin*, *cataract*, *coumadin*, *insulin*, *heparin*, and *pleural* from the raw nursing text. Figure 5 shows the coherence score comparison of LDA and NMF models with the number of topics varying from 2 to 500.

3 ICD-9 Code Group Prediction

ICD-9 codes are a taxonomy of diagnostic codes typically used by healthcare professionals and insurers when discussing medical conditions. This study only focuses on category-level (group) predictions, owing to the high granularity of the diagnostic codes. Each code group comprises a set of similar diseases, and most of the health conditions can be categorized into a unique group. This study focuses on the risk stratification as a multi-label problem, where each nursing note is mapped to multiple ICD-9 code groups. The ICD-9 codes for a given admission are mapped into 19 distinct code groups[3]. Note that the ICD-9 code range of 760 – 779 was left out since it corresponds to the *conditions originating in the perinatal period*, which is usually assigned to newborns, who are excluded from this study as per the defined cohort selection criteria (see Section 2.1). Additionally, to lower the computational cost of training, we merged all the reference and supplemental V-codes into a single code group. Figure 6 presents a spider plot depicting the statistics of the ratio of the number of patients in a particular code group to the total number of patients in the cohort. Although our work and the state-of-the-art (Purushotham et al., 2018) differ in data and cohort selection, it can be observed from Figure 6 that both the works share similar statistics concerning the ICD-9 code groups, thus facilitating a fair comparison of performance.

3.1 Deep Neural Architectures

We used two deep neural architectures, Multilayer Perceptron (MLP) and Attention-based LSTM (A-LSTM), for the multi-label ICD-9 code group prediction task. The deep models were trained to minimize a binary cross-entropy loss function using an Adam optimizer, with a batch size of 128, for eight epochs.

3.1.1 Multi-Layer Perceptron

The MLP is a feed-forward artificial neural network consisting of multiple layers of neurons (nodes) interacting using weighted connections.

[3] http://tdrdata.com/ipd/ipd_SearchForICD9CodesAndDescriptions.aspx.

MLP offers several advantages including adaptive learning, fault tolerance, parallelism, and generalizability. The output of a neuron in every layer serves as an input to the subsequent layer. A neuron in the current layer (l) with the input $I^{(l)}$ is activated in the following layer ($l+1$) as $g^{(l)}(W^{(l)} \cdot I^{(l)} + b^{(l)})$, where $g^{(l)}$ is a non-linear activation such as Rectified Linear Unit (ReLU), tanh, or logistic sigmoid, and $b^{(l)}$ and $W^{(l)}$ are the bias and weight matrix at layer l. MLP uses backpropagation to determine the gradient of the loss function needed to learn an optimal set of weights and biases needed to minimize a loss function. This study employs an MLP network with one hidden layer of 75 nodes, activated using a ReLU function, and one output layer of 19 nodes, activated using a sigmoid function.

3.1.2 Attention-based LSTM

The LSTM effectively captures the long-term dependencies and overcomes the gradient vanishing problem which is crucial in the accurate risk stratification using unstructured nursing notes. LSTMs introduce an adaptive gating mechanism to determine the extent to which the LSTM memory units must retain the previous state (c_{t-1}) and memorize the features in the current state (c_t). Typically, four gates composite an LSTM network including the input gate i, the forget gate f, the output gate o, and the candidate value g for the cell state. The precise form of an LSTM update at a layer l and time step t is computed as:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^{(l)} \begin{pmatrix} h_{t-1}^{(l)} \\ h_t^{(l-1)} \end{pmatrix} \quad (4)$$

$$c_t^{(l)} = f \odot c_{t-1}^{(l)} + i \odot g \quad (5)$$

$$h_t^{(l)} = o \odot \tanh(c_t^{(l)}) \quad (6)$$

where $\odot$ denotes element-wise multiplication, h_t is the output at a time step t, and $W^{(l)}$ is a $[4n \times 2n]$ weight matrix at layer l.

Attentive neural models have been successfully applied to several NLP tasks including sentence summarization, text entailment, and reading comprehension (Bahdanau et al., 2014). This study utilizes the attention mechanism for the clinical task of risk stratification as ICD-9 code group prediction. Let H be the matrix of output vectors $[h_1, h_2, \ldots, h_T]$ produced from LSTM. The representation r_j of a nursing note η_j after T time steps is computed as $H \cdot (\text{softmax}(v^T \cdot \tanh(H)))^T$, where v is a trainable parameter. This study utilizes an attention-based LSTM with dimension size of 289 for the embedding (17 time steps) and 300 for the LSTM hidden state. The multi-label classification is facilitated using a sigmoid activation of the final A-LSTM output.

3.2 Experimental Results and Discussion

To experimentally validate the proposed approach, we performed an exhaustive benchmarking on the clinical nursing notes obtained from the MIMIC-III database. The experiments were performed using a server running Ubuntu OS with 56 cores of Intel Xeon processors, 128 GB RAM, 3 TB hard drive, and two NVIDIA Tesla M40 GPUs. A significant challenge arose due to the manifold nature of diseases, as each patient record was assigned a set of ICD-9 code groups. This study employs a pair-wise comparison of the actual and predicted code group sets. Five standard evaluation metrics including Accuracy (ACC), F1 score, MCC score, Area Under the Precision-Recall Curve (AUPRC), and Area Under the ROC Curve (AUROC) were employed to evaluate the performance of the proposed coherence-based modeling approaches, classified using MLP and A-LSTM. Ten-fold cross-validation was performed to assess the predictability of the proposed models. Table 1 tabulates the performance of the proposed modeling approaches using the proposed *FarSight* approach for data aggregation along with two standard baselines. We observe that the proposed C-LDA model outperforms the C-NMF model in accurately classifying the diagnostic ICD-9 code groups. Additionally, from Table 1, we observe that the proposed C-LDA model outperforms the other standard baselines including LDA and NMF without coherence scores.

AUPRC varies with the change in the ratio of the target classes in the data and hence is more informative than AUROC while evaluating imbalanced data (Saito and Rehmsmeier, 2015). F1 score captures both precision and recall of the prediction, and MCC score takes into account, the true positives, false positives, and false negatives, thus serving as a balanced measure even with class imbalance. Due to the significant class imbalance in the underlying corpus (see Figure 6), AUROC and MCC scores serve as accurate evaluation metrics. The existing works, including the state-of-

the-art model (Purushotham et al., 2018), are built on the structured nature of the EHRs, modeled using numerical feature sets (e.g., lab results) to aid in the prediction of clinical events. From Figure 7, we remark that the proposed approach built on the unstructured medical text and preprocessed using the *FarSight* approach outperformed the state-of-the-art model by 11.50% in AUPRC and 1.16% in AUROC. Furthermore, the existing works do not benchmark their performance on metrics other than AUPRC and AUROC. We urge that the other metrics presented in this study aid in the accurate assessment of the proposed models, essential in determining the reliability of the underlying CDSS. It can also be noted that the *FarSight* approach effectively models the unstructured data to facilitate the detection of the onset of the disease with the earliest recorded symptoms, and such modeling results in an improvement in the clinical decision-making process. We observe that utilizing the proposed approach leads to accurate health risk appraisal well in advance, with an overall accuracy of 80%. Thus, CDSSs built on the predictive capabilities of *FarSight*-aggregated and C-LDA classified modeling could demonstrate effective patient-centric and evidence-based risk assessment, thus ensuring proper channeling of preventive and prioritized care.

3.3 Sensitivity Analysis

The experimental results in Table 1 highlight the efficacy of the proposed models over the state-of-the-art model (see Figure 7) and standard baselines, including LDA and NMF without coherence scores, in modeling the raw patient-specific clinical nursing notes. To analyze the significance of the observed performance further, we performed a statistical sensitivity analysis. Sensitivity analysis

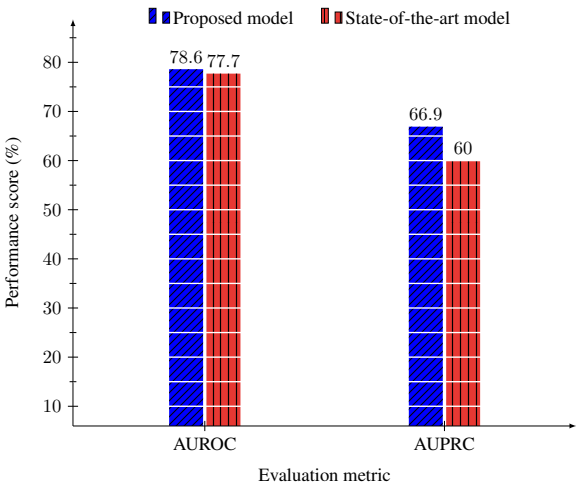


Figure 7: Comparison of the proposed approach with the state-of-the-art model (Purushotham et al., 2018).

(Simar and Wilson, 1998) is a potential approach that facilitates decision-making by measuring the extent to which the optimal solution is sensitive to the change in the input of one or more parameters.

To understand the distribution of the underlying data, we employed the Kolmogorov-Smirnov test for normality (J and Jr, 1951), which revealed that the data was not normally distributed. The performance of an algorithm measured as a result of ten-fold cross-validation forms the treatment population of that approach. Additionally, note that each sample (performance score) in the corresponding treatments of the algorithms under comparison utilize the same k^{th} fold data, and thus the samples are generated as a function of the same input population. Therefore, to perform the sensitivity analysis, we employed a nonparametric paired samples Wilcoxon signed-rank test (Wilcoxon, 1992) at a significance level (α) of 5%. The null hypothesis in Wilcoxon signed-rank test is that the two treatments are drawn from the same distribution,

Data Model	Classifier	Performance score				
		ACC	F1	MCC	AUPRC	AUROC
C-LDA $(140, 792 \times 100)$	MLP	$\mathbf{0.7954 \pm 0.0003}$	0.7175 ± 0.0008	$\mathbf{0.5743 \pm 0.0006}$	$\mathbf{0.6692 \pm 0.0006}$	$\mathbf{0.7857 \pm 0.0004}$
	A-LSTM	0.7932 ± 0.0002	$\mathbf{0.7186 \pm 0.0002}$	0.5712 ± 0.0007	0.6660 ± 0.0007	0.7854 ± 0.0013
C-NMF $(140, 792 \times 100)$	MLP	0.7826 ± 0.0004	0.7011 ± 0.0008	0.5480 ± 0.0007	0.6530 ± 0.0013	0.7735 ± 0.0006
	A-LSTM	0.7811 ± 0.0005	0.6990 ± 0.0040	0.5449 ± 0.0007	0.6510 ± 0.0009	0.7715 ± 0.0026
LDA $(140, 792 \times 100)$	MLP	0.7950 ± 0.0003	0.7168 ± 0.0020	0.5735 ± 0.0012	0.6685 ± 0.0013	0.7848 ± 0.0011
	A-LSTM	0.7930 ± 0.0007	0.7153 ± 0.0034	0.5701 ± 0.0022	0.6655 ± 0.0013	0.7833 ± 0.0020
NMF $(140, 792 \times 100)$	MLP	0.7829 ± 0.0006	0.7029 ± 0.0016	0.5498 ± 0.0009	0.6530 ± 0.0017	0.7744 ± 0.0007
	A-LSTM	0.7815 ± 0.0008	0.6935 ± 0.0052	0.5451 ± 0.0024	0.6535 ± 0.0014	0.7689 ± 0.0031

Table 1: Experimental results for ICD-9 code group prediction using MLP and A-LSTM.

Data Model	Classifier	ACC		F1		MCC		AUPRC		AUROC	
		p	z	p	z	p	z	p	z	p	z
C-LDA $(140,792 \times 100)$	MLP	–		0.005	−2.803	–		–		–	
	A-LSTM	0.005	−2.803	–		0.005	−2.803	0.005	−2.803	0.009	−2.599
C-NMF $(140,792 \times 100)$	MLP	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803
	A-LSTM	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803
LDA $(140,792 \times 100)$	MLP	0.005	−2.803	0.009	−2.599	0.007	−2.701	0.016	−2.395	0.009	−2.599
	A-LSTM	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803
NMF $(140,792 \times 100)$	MLP	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803
	A-LSTM	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803	0.005	−2.803

Table 2: A paired samples Wilcoxon signed-rank test (two-tailed, $p < 0.05$) for the proposed model with the best performance against other modeling strategies.

which is rejected in favor of the alternate hypothesis when the significance level (p-value) resulting from the test is higher than the preset α. Table 2 presents the results of our sensitivity analysis for the proposed model with the best performance against other modeling strategies. From Table 2, it can be observed that the value of p is always lower than the preset α of 0.05. Thus, we conclude that the proposed model with the best performance is statistically significant than the other approaches and baseline methods with respect to all the employed performance evaluation metrics.

4 Concluding Remarks

In this paper, we presented *FarSight*, a technique for detecting the onset of the disease with the earliest recorded symptoms and infections, to provide preventive and prioritized care, in turn aiding in the reduction of the morbidity rate. Two coherence-based topic modeling approaches were employed to capture the semantic information in the nursing notes and derive the optimal data representations with emphasis on the human interpretability of the derived clinical concepts. The obtained data representations were effectively leveraged for diagnostic ICD-9 code group prediction using deep neural architectures. Unlike in the previous works, we benchmarked the performance of our proposed models using several evaluation metrics which are essential in the accurate assessment of the reliability of the models. The proposed model captured the valuable patient-specific information present in the informally written nursing notes and outperformed the structured EHR data based state-of-the-art model with an improvement of 11.50% in terms of AUPRC and 1.16% in terms of AUROC. Furthermore, we also observed that the proposed *FarSight*-aggregated and C-LDA classified model captured the discriminative features of the nursing notes and consistently outperformed several other standard models, including C-NMF, LDA, and NMF. Moreover, our model eliminates the dependency on structured EHRs for the development of CDSSs and is extremely vital in countries with low EHR adoption rates.

Although the proposed approach effectively stratifies the patients' risk and the associated complications, it can be enhanced further, which calls for further research on this topic. First, the proposed approach only models the unstructured nursing text and neglects the structured EHR information (e.g., lab results), which can potentially be utilized to facilitate robust patient profiling. Second, the modeling presented in this study does not account for real-time clinical data. In the future, we intend on exploring the techniques for modeling structured EHR data along with the data modeled from the unstructured clinical nursing notes. We also aim at validating our model on real-time clinical data to enhance its predictability and adaptability, thus focusing on the need for time-aware, dependable architectures in real-world hospital scenarios.

Acknowledgments

This work is funded by the Government of India's DST-SERB Early Career Research Grant (ECR/2017/001056) to Sowmya Kamath S. Any opinions, findings, and recommendations or conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agency.

References

Nikolaos Aletras and Mark Stevenson. 2013. Evaluating Topic Coherence Using Distributional Semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 13–22, Potsdam, Germany. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Tal Baumel, Jumana Nassour-Kassis, Raphael Cohen, Michael Elhadad, and Noémie Elhadad. 2018. Multi-Label Classification of Patient Notes: Case Study on ICD Code Assignment. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. In *From Form to Meaning: Processing Texts Automatically, Proceedings of the Biennial GSCL Conference 2009*, volume Normalized, pages 31–40, Tübingen.

Gilles Clermont, Derek C Angus, Stephen M DiRusso, Martin Griffin, and Walter T Linde-Zwirble. 2001. Predicting hospital mortality for patients in the intensive care unit: A comparison of artificial neural networks with logistic regression models. *Critical care medicine*, 29(2):291–296.

GS Doig, KJ Inman, WJ Sibbald, CM Martin, and JM Robertson. 1993. Modeling mortality in the intensive care unit: comparing the performance of a back-propagation, associative-learning neural network with multivariate logistic regression. *Proceedings. Symposium on Computer Applications in Medical Care*, pages 361–365.

Sebastien Dubois, Nathanael Romano, David C Kale, Nigam Shah, and Kenneth Jung. 2017. Learning Effective Representations from Clinical Notes. *arXiv preprint arXiv:1705.07025*.

Keith Feldman, Nicholas Hazekamp, and Nitesh V Chawla. 2016. Mining the Clinical Narrative: All Text are Not Equal. In *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 271–280. IEEE.

Jim Grigsby, Robert Kooken, and John Hershberger. 1994. Simulated neural networks to predict outcomes, costs, and length of stay among orthopedic rehabilitation patients. *Archives of physical medicine and rehabilitation*, 75(10):1077–1081.

C William Hanson and Bryan E Marshall. 2001. Artificial intelligence applications in the intensive care unit. *Critical care medicine*, 29(2):427–435.

Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, and Aram Galstyan. 2017. Multitask learning and benchmarking with clinical time series data. *arXiv preprint arXiv:1703.07771*.

Vijay Huddar, Bapu Koundinya Desiraju, Vaibhav Rajan, Sakyajit Bhattacharya, Shourya Roy, and Chandan K Reddy. 2016. Predicting Complications in Critical Care Using Heterogeneous Clinical Data. *IEEE Access*, 4:7988–8001.

Frank J and Massey Jr. 1951. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American statistical Association*, 46(253):68–78.

Yohan Jo, Natasha Loghmanpour, and Carolyn Penstein Rosé. 2015. Time Series Analysis of Nursing Notes for Mortality Prediction via a State Transition Topic Model. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1171–1180, New York, NY, USA. ACM.

Alistair EW Johnson, Tom J Pollard, and Roger G Mark. 2017. Reproducibility in critical care: a mortality prediction case study. In *Proceedings of the 2nd Machine Learning for Healthcare Conference*, volume 68 of *Proceedings of Machine Learning Research*, pages 361–376, Boston, Massachusetts. PMLR.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3:160035.

Daniel D. Lee and H. Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 535–541, Cambridge, MA, USA. MIT Press.

Prabha Susy Mathew and Anitha S Pillai. 2015. Big data solutions in healthcare: Problems and perspectives. In *2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pages 1–6. IEEE.

Romain Pirracchio. 2016. *Mortality Prediction in the ICU Based on MIMIC-II Results from the Super ICU Learner Algorithm (SICULA) Project*, pages 295–313. Springer International Publishing, Cham.

Sanjay Purushotham, Chuizheng Meng, Zhengping Che, and Yan Liu. 2018. Benchmarking deep learning models on large healthcare datasets. *Journal of Biomedical Informatics*, 83:112–134.

Michael Röder, Andreas Both, and Alexander Hinneburg. 2015. Exploring the Space of Topic Coherence Measures. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 399–408, New York, NY, USA. ACM.

Takaya Saito and Marc Rehmsmeier. 2015. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*, 10(3):1–21.

Léopold Simar and Paul W Wilson. 1998. Sensitivity Analysis of Efficiency Scores: How to Bootstrap in Nonparametric Frontier Models. *Management science*, 44(1):49–61.

Kathleen B To and Lena M Napolitano. 2012. Common Complications in the Critically Ill Patient. *Surgical Clinics*, 92(6):1519–1557.

Jack V Tu and Michael RJ Guerriere. 1993. Use of a Neural Network as a Predictive Instrument for Length of Stay in the Intensive Care Unit Following Cardiac Surgery. *Computers and Biomedical Research*, 26(3):220–229.

Yanshan Wang, Naveed Afzal, Sunyang Fu, Liwei Wang, Feichen Shen, Majid Rastegar-Mojarad, and Hongfang Liu. 2018. MedSTS: a resource for clinical semantic textual similarity. *Language Resources and Evaluation*.

Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In Samuel Kotz and Norman L Johnson, editors, *Breakthroughs in Statistics: Methodology and Distribution*, pages 196–202. Springer New York, New York, NY.

Christine M Wollschlager and Arnold R Conrad. 1988. Common complications in critically ill patients. *Disease-a-Month*, 34(5):225–293.

Aaron Zalewski, William Long, Alistair EW Johnson, Roger G Mark, and H Lehman Li-wei. 2017. Estimating patient's health state using latent structure inferred from clinical time series and text. In *2017 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, pages 449–452. IEEE.

Predicting the Role of Political Trolls in Social Media

Atanas Atanasov
Sofia University
Bulgaria
amitkov@uni-sofia.bg

Gianmarco De Francisci Morales
ISI Foundation
Italy
gdfm@acm.org

Preslav Nakov
Qatar Computing Research Institute, HBKU, Qatar
pnakov@qf.org.qa

Abstract

We investigate the political roles of "Internet trolls" in social media. Political trolls, such as the ones linked to the Russian Internet Research Agency (IRA), have recently gained enormous attention for their ability to sway public opinion and even influence elections. Analysis of the online traces of trolls has shown different behavioral patterns, which target different slices of the population. However, this analysis is manual and labor-intensive, thus making it impractical as a first-response tool for newly-discovered troll farms. In this paper, we show how to automate this analysis by using machine learning in a realistic setting. In particular, we show how to classify trolls according to their political role —left, news feed, right— by using features extracted from social media, i.e., Twitter, in two scenarios: (*i*) in a traditional supervised learning scenario, where labels for trolls are available, and (*ii*) in a distant supervision scenario, where labels for trolls are not available, and we rely on more-commonly-available labels for news outlets mentioned by the trolls. Technically, we leverage the community structure and the text of the messages in the online social network of trolls represented as a graph, from which we extract several types of learned representations, i.e., embeddings, for the trolls. Experiments on the "IRA Russian Troll" dataset show that our methodology improves over the state-of-the-art in the first scenario, while providing a compelling case for the second scenario, which has not been explored in the literature thus far.

1 Introduction

Internet "trolls" are users of an online community who quarrel and upset people, seeking to sow discord by posting inflammatory content. More recently, organized "troll farms" of political opinion manipulation trolls have also emerged. Such farms usually consist of state-sponsored agents who control a set of pseudonymous user accounts and personas, the so-called "sockpuppets", which disseminate misinformation and propaganda in order to sway opinions, destabilize the society, and even influence elections (Linvill and Warren, 2018).

The behavior of political trolls has been analyzed in different recent circumstances, such as the 2016 US Presidential Elections and the Brexit referendum in UK (Linvill and Warren, 2018; Llewellyn et al., 2018). However, this kind of analysis requires painstaking and time-consuming manual labor to sift through the data and to categorize the trolls according to their actions. Our goal in the current paper is to automate this process with the help of machine learning (ML). In particular, we focus on the case of the 2016 US Presidential Elections, for which a public dataset from Twitter is available. For this case, we consider only accounts that post content in English, and we wish to divide the trolls into some of the functional categories identified by Linvill and Warren (2018): *left troll*, *right troll*, and *news feed*.

We consider two possible scenarios. The first, prototypical ML scenario is supervised learning, where we want to learn a function from users to categories {*left*, *right*, *news feed*}, and the ground truth labels for the troll users are available. This scenario has been considered previously in the literature by Kim et al. (2019). Unfortunately, a solution for such a scenario is not directly applicable to a real-world use case. Suppose a new troll farm trying to sway the upcoming European or US elections has just been discovered. While the identities of the accounts might be available, the labels to learn from would not be present. Thus, any supervised machine learning approach would fall short of being a fully automated solution to our initial problem.

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 1023–1034
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

A more realistic scenario assumes that labels for troll accounts are *not available*. In this case, we need to use some external information in order to learn a labeling function. Indeed, we leverage more persistent entities and their labels: news media. We assume a learning scenario with distant supervision where labels for news media are available. By combining these labels with a citation graph from the troll accounts to news media, we can infer the final labeling on the accounts themselves without any need for manual labeling.

One advantage of using distant supervision is that we can get insights about the behavior of a newly-discovered troll farm quickly and effortlessly. Differently from troll accounts in social media, which usually have a high churn rate, news media accounts in social media are quite stable. Therefore, the latter can be used as an anchor point to understand the behavior of trolls, for which data may not be available.

We rely on embeddings extracted from social media. In particular, we use a combination of embeddings built on the user-to-user mention graph, the user-to-hashtag mention graph, and the text of the tweets of the troll accounts. We further explore several possible approaches using label propagation for the distant supervision scenario.

As a result of our approach, we improve the classification accuracy by more than 5 percentage points for the supervised learning scenario. The distant supervision scenario has not previously been considered in the literature, and is one of the main contributions of the paper. We show that even by hiding the labels from the ML algorithm, we can recover 78.5% of the correct labels.

The contributions of this paper can be summarized as follows:

- We predict the political role of Internet trolls (*left*, *news feed*, *right*) in a realistic, unsupervised scenario, where labels for the trolls are not available, and which has not been explored in the literature before.
- We propose a novel distant supervision approach for this scenario, based on graph embeddings, BERT, and label propagation, which projects the more-commonly-available labels for news media onto the trolls who cited these media.
- We improve over the state of the art in the traditional, fully supervised setting, where training labels are available.

2 Related Work

2.1 Trolls and Opinion Manipulation

The promise of social media to democratize content creation (Kaplan and Haenlein, 2010) has been accompanied by many malicious attempts to spread misleading information over this new medium, which quickly got populated by *sockpuppets* (Kumar et al., 2017), *Internet water army* (Chen et al., 2013), *astroturfers* (Ratkiewicz et al., 2011), and *seminar users* (Darwish et al., 2017). Several studies have shown that trust is an important factor in online relationships (Ho et al., 2012; Ku, 2012; Hsu et al., 2014; Elbeltagi and Agag, 2016; Ha et al., 2016), but building trust is a long-term process and our understanding of it is still in its infancy (Salo and Karjaluoto, 2007). This makes it easy for politicians and companies to manipulate user opinions in community forums (Dellarocas, 2006; Li et al., 2016; Zhuang et al., 2018).

Trolls. Social media have seen the proliferation of fake news and clickbait (Hardalov et al., 2016; Karadzhov et al., 2017a), aggressiveness (Moore et al., 2012), and trolling (Cole, 2015). The latter often is understood to concern malicious online behavior that is intended to disrupt interactions, to aggravate interacting partners, and to lure them into fruitless argumentation in order to disrupt online interactions and communication (Chen et al., 2013). Here we are interested in studying not just any trolls, but those that engage in opinion manipulation (Mihaylov et al., 2015a,b, 2018). This latter definition of *troll* has also become prominent in the general public discourse recently. Del Vicario et al. (2016) have also suggested that the spreading of misinformation online is fostered by the presence of polarization and echo chambers in social media (Garimella et al., 2016, 2017, 2018).

Trolling behavior is present and has been studied in all kinds of online media: online magazines (Binns, 2012), social networking sites (Cole, 2015), online computer games (Thacker and Griffiths, 2012), online encyclopedia (Shachaf and Hara, 2010), and online newspapers (Ruiz et al., 2011), among others.

Troll detection has been addressed by using domain-adapted sentiment analysis (Seah et al., 2015), various lexico-syntactic features about user writing style and structure (Chen et al., 2012; Mihaylov and Nakov, 2016), and graph-based approaches over signed social networks (Kumar et al., 2014).

Sockpuppet is a related notion, and refers to a person who assumes a false identity in an Internet community and then speaks to or about themselves while pretending to be another person. The term has also been used to refer to opinion manipulation, e.g., in Wikipedia (Solorio et al., 2014). Sockpuppets have been identified by using authorship-identification techniques and link analysis (Bu et al., 2013). It has been also shown that sockpuppets differ from ordinary users in their posting behavior, linguistic traits, and social network structure (Kumar et al., 2017).

Internet Water Army is a literal translation of the Chinese term *wangluo shuijun*, which is a metaphor for a large number of people who are well organized to flood the Internet with purposeful comments and articles. Internet water army has been allegedly used in China by the government (also known as *50 Cent Party*) as well as by a number of private organizations.

Astroturfing is an effort to simulate a political grass-roots movement. It has attracted strong interest from political science, and research on it has focused on massive streams of microblogging data (Ratkiewicz et al., 2011).

Identification of malicious accounts in social media includes detecting spam accounts (Almaatouq et al., 2016; McCord and Chuah, 2011), fake accounts (Fire et al., 2014; Cresci et al., 2015), compromised and phishing accounts (Adewole et al., 2017). Fake profile detection has also been studied in the context of cyber-bullying (Galán-García et al., 2016). A related problem is that of *Web spam detection*, which has been addressed as a text classification problem (Sebastiani, 2002), e.g., using spam keyword spotting (Dave et al., 2003), lexical affinity of arbitrary words to spam content (Hu and Liu, 2004), frequency of punctuation and word co-occurrence (Li et al., 2006).

Trustworthiness and veracity analytics of online statements is an emerging research direction, especially given the recent interest in fake news (Lazer et al., 2018). It is related to trolls, as they often engage in opinion manipulation and rumor spreading (Vosoughi et al., 2018). Research topics include predicting the credibility of information in social media (Ma et al., 2016; Mitra et al., 2017; Karadzhov et al., 2017b; Popat et al., 2017) and political debates (Hassan et al., 2015; Gencheva et al., 2017; Jaradat et al., 2018), as well as stance classification (Mohtarami et al., 2018). For example, Castillo et al. (2011) leverage user reputation, author writing style, and various time-based features, Canini et al. (2011) analyze the interaction of content and social network structure, and Morris et al. (2012) studied how Twitter users judge truthfulness. Zubiaga et al. (2016) study how people handle rumors in social media, and found that users with higher reputation are more trusted, and thus can spread rumors easily. Lukasik et al. (2015) use temporal patterns to detect rumors and to predict their frequency, and Zubiaga et al. (2016) focus on conversational threads. More recent work has focused on the credibility and the factuality in community forums (Nakov et al., 2017; Mihaylova et al., 2018, 2019; Mihaylov et al., 2018).

2.2 Understanding the Role of Political Trolls

None of the above work has focused on understanding the role of political trolls. The only closely relevant work is that of Kim et al. (2019), who predict the roles of the Russian trolls on Twitter by leveraging social theory and Actor-Network Theory approaches. They characterize trolls using the digital traces they leave behind, which is modeled using a time-sensitive semantic edit distance. For this purpose, they use the "IRA Russian Troll" dataset (Linvill and Warren, 2018), which we also use in our experiments. However, we have a very different approach based on graph embeddings, which we show to be superior to their method in the supervised setup. We further experiment with a new, and arguably more realistic, setup based on distant supervision, where labels are not available. To the best of our knowledge, this setup has not been explored in previous work.

2.3 Graph Embeddings

Graph embeddings are machine learning techniques to model and capture key features from a graph automatically. They can be trained either in a supervised or in an unsupervised manner (Cai et al., 2018). The produced embeddings are latent vector representations that map each vertex V in a graph G to a d-dimensional vector. The vectors capture the underlying structure of the graph by putting "similar" vertices close together in the vector space. By expressing our data as a graph structure, we can leverage and extract critical insights about the topology and the contextual relationships between the vertices in the graph.

In mathematical terms, graph embeddings can be expressed as a function $f : V \rightarrow R^d$ from the set of vertices V to a set of embeddings, where d is the dimensionality of the embeddings. The function f can be represented as a matrix of dimensions $|V| \times d$. In our experiments, we train Graph Embeddings in an unsupervised manner by using node2vec (Grover and Leskovec, 2016), which is based on random walks over the graph. Essentially, this is an application of the well-known skip-gram model (Mikolov et al., 2013) from word2vec to random walks on graphs.

Besides *node2vec*, there have been a number of competing proposals for building graph embeddings; see (Cai et al., 2018) for an extensive overview of the topic. For example, *SNE* (Liao et al., 2018) model both the graph structure and some node attributes. Similarly, *Line* (Tang et al., 2015) represent each node as the concatenation of two embedded vectors that model first- and second-order proximity. *TriDNR* (Pan et al., 2016) represents nodes by coupling several neural network models. For our experiments, we use node2vec, as we do not have access to user attributes: the users have been banned from Twitter, their accounts were suspended, and we only have access to their tweets thanks to the "IRA Russian Trolls" dataset.

3 Method

Given a set of known political troll users (each user being represented as a collection of their tweets), we aim to detect their role: *left*, *right*, or *news feed*. Linvill and Warren (2018) describe these roles as follows:

Right Trolls spread nativist and right-leaning populist messages. Such trolls support the candidacy and Presidency of Donald Trump and denigrate the Democratic Party; moreover, they often send divisive messages about mainstream and moderate Republicans.

Left Trolls send socially liberal messages and discuss gender, sexual, religious, and -especially- racial identity. Many tweets are seemed intentionally divisive, attacking mainstream Democratic politicians, particularly Hillary Clinton, while supporting Bernie Sanders prior to the elections.

News Feed Trolls overwhelmingly present themselves as US local news aggregators, linking to legitimate regional news sources and tweeting about issues of local interest.

Technically, we leverage the community structure and the text of the messages in the social network of political trolls represented as a graph, from which we learn and extract several types of vector representations, i.e., troll user embeddings. Then, armed with these representations, we tackle the following tasks:

T1 A fully supervised learning task, where we have labeled training data with example troll and their roles;

T2 A distant supervision learning task, in which labels for the troll roles are *not* available at training time, and thus we use labels for news media as a proxy, from which we infer labels for the troll users.

3.1 Embeddings

We use two graph-based (user-to-hashtag and user-to-mentioned-user) and one text-based (BERT) embedding representations.

3.1.1 U2H

We build a bipartite, undirected User-to-Hashtag (U2H) graph, where nodes are users and hashtags, and there is an edge (u, h) between a user node u and a hashtag node h if user u uses hashtag h in their tweets. This graph is bipartite as there are no edges connecting two user nodes or two hashtag nodes. We run node2vec (Grover and Leskovec, 2016) on this graph, and we extract the embeddings for the users (we ignore the hashtag embeddings). We use 128 dimensions for the output embeddings. These embeddings capture how similar troll users are based on their usage of hashtags.

3.1.2 U2M

We build an undirected User-to-Mentioned-User (U2M) graph, where the nodes are users, and there is an edge (u, v) between two nodes if user u mentions user v in their tweets (i.e., u has authored a tweet that contains "@v"). We run node2vec on this graph and we extract the embeddings for the users. As we are interested only in the troll users, we ignore the embeddings of users who are only mentioned by other trolls. We use 128 dimensions for the output embeddings. The embeddings extracted from this graph capture how similar troll users are according to the targets of their discussions on the social network.

3.1.3 BERT

BERT offers state-of-the-art text embeddings based on the Transformer architecture (Devlin et al., 2019). We use the pre-trained BERT-large, uncased model, which has 24-layers, 1024-hidden, 16-heads, and 340M parameters, which yields output embeddings with 768 dimensions. Given a tweet, we generate an embedding for it by averaging the representations of the BERT tokens from the penultimate layer of the neural network. To obtain a representation for a user, we average the embeddings of all their tweets. The embeddings extracted from the text capture how similar users are according to their use of language.

3.2 Fully Supervised Learning (T1)

Given a set of troll users for which we have labels, we use the above embeddings as a representation to train a classifier. We use an L2-regularized logistic regression (LR) classifier. Each troll user is an example, and the label for the user is available for training thanks to manual labeling. We can therefore use cross-validation to evaluate the predictive performance of the model, and thus the predictive power of the features.

We experiment with two ways of combining features: *embedding concatenation* and *model ensembling*. Embedding concatenation concatenates the feature vectors from different embeddings into a longer feature vector, which we then use to train the LR model. Model ensembling instead trains a separate model with each kind of embedding, and then merges the prediction of the different models by averaging the posterior probabilities for the different classes. Henceforth, we denote embedding concatenation with the symbol $\|$ and model ensembling with $\oplus$. For example, U2H $\|$ U2M is a model trained on the concatenation of U2H and U2M embeddings, while U2H $\oplus$ BERT represents the average predictions of two models, one trained on U2H embeddings and one on BERT.

3.3 Distant Supervision (T2)

In the distant supervision scenario, we assume not to have access to user labels. Given a set of troll users without labels, we use the embeddings described in Section 3.1 together with mentions of *news media* by the troll users to create proxy models. We assume that labels for news media are readily available, as they are stable sources of information that have a low churn rate.

We propagate labels from the given media to the troll user that mentions them according to the following media-to-user mapping:

$$\begin{aligned} \textit{LEFT} &\rightarrow \textit{left} \\ \textit{RIGHT} &\rightarrow \textit{right} \\ \textit{CENTER} &\rightarrow \textit{news feed} \end{aligned} \tag{1}$$

This propagation can be done in different ways: (a) by training a proxy model for media and then applying it to users, (b) by additionally using label propagation (LP) for semi-supervised learning.

Let us describe the proxy model propagation for (a) first. Let M be the set of media, and U be the set of users. We say a user $u \in U$ mentions a medium $m \in M$ if u posts a tweet that contains a link to the website of m. We denote the set of users that mention the medium m as $C_m \subseteq U$.

We can therefore create a representation for a medium by aggregating the embeddings of the users that mention the target medium. Such a representation is convenient as it lies in the same space as the user representation. In particular, given a medium $m \in M$, we compute its representation $R(m)$ as

$$R(m) = \frac{1}{|C_m|} \sum_{u \in C_m} R(u), \tag{2}$$

where $R(u)$ is the representation of user u, i.e., one (or a concatenation) of the embeddings described in Section 3.1.

Finally, we can train a LR model that uses $R(m)$ as features and the label for the medium $l(m)$. This model can be applied to predict the label of a user u by using the same type of representation $R(u)$, and the label mapping in Equation 1.

Label Propagation (b) is a transductive, graph-based, semi-supervised machine learning algorithm that, given a small set of labeled examples, assigns labels to previously unlabeled examples. The labels of each example change in relationship to the labels of *neighboring* ones in a properly-defined graph.

More formally, given a partially-labeled dataset of examples $X = X_u \cup X_l$, of which X_l are labeled examples with labels Y_l, and X_u are unlabeled examples, and a similarity graph $G(X, E)$, the label propagation algorithm finds the set of unknown labels Y_u such that the number of discordant pairs $(u, v) \in E : y_u \neq y_v$ is minimized, where y_z is the label assigned to example z.

Role	Users	Tweets	User Example	Tweet Example
Left	233	427 141	@samirgooden	@MichaelSkolnik @KatrinaPierson @samesfandiari Trump folks need to stop going on CNN.
Right	630	711 668	@chirrmorre	BREAKING: Trump ERASES Obama's Islamic Refugee Policy! https://t.co/uPTneTMNM5
News Feed	54	598 226	@dailysandiego	Exit poll: Wisconsin GOP voters excited, scared about Trump #politics

Table 1: Statistics and examples from the IRA Russian Trolls Tweets dataset.

The algorithm works as follows: At every iteration of propagation, each unlabeled node updates its label to the most frequent one among its neighbors. LP reaches convergence when each node has the same label as the majority of its neighbors. We define two different versions of LP by creating two different versions of the similarity graph G.

LP1 *Label Propagation using direct mention.*
In the first case, the set of edges among users U in the similarity graph G consists of the logical OR between the 2-hop closure of the U2H and the U2M graph. That is, for each two users $u, v \in U$, there is an edge in the similarity graph $(u, v) \in E$ if u and v share a common hashtag or a common user mention

$$(u, h) \in \text{U2H} \wedge (v, h) \in \text{U2H} \vee (u, w) \in \text{U2M} \wedge (v, w) \in \text{U2M}$$

The graph therefore uses the same information that is available to the embeddings.

To this graph, which currently encompasses only the set of users U, we add connections to the set of media M. We add an edge between each pair (u, m) if $u \in C_m$. Then, we run the label propagation algorithm, which propagates the labels from the labeled nodes M to the unlabeled nodes U, thanks to the mapping from Equation 1.

LP2 *Label Propagation based on a similarity graph.*
In this case, we use the same representation for the media as in the proxy model case above, as described by Equation 2. Then, we build a similarity graph among media and users based on their embeddings. For each pair $x, y \in U \cup M$ there is an edge in the similarity graph $(x, y) \in E$ iff

$$\text{sim}(R(x), R(y)) > \tau,$$

where sim is a similarity function between vectors, e.g., cosine similarity, and τ is a user-specified parameter that regulates the sparseness of the similarity graph.

Finally, we perform label propagation on the similarity graph defined by the embedding similarity, with the set of nodes corresponding to M starting with labels, and with the set of nodes corresponding to U starting without labels.

4 Data

4.1 IRA Russian Troll Tweets

Our main dataset contains 2 973 371 tweets by 2848 Twitter users, which the US House Intelligence Committee has linked to the Russian Internet Research Agency (IRA). The data was collected and published by Linvill and Warren (2018), and then made available online.[1] The time span covers the period from February 2012 to May 2018.

The trolls belong to the following manually assigned roles: Left Troll, Right Troll, News Feed, Commercial, Fearmonger, Hashtag Gamer, Non English, Unknown. Kim et al. (2019) have argued that the first three categories are not only the most frequent, but also the most interesting ones. Moreover, focusing on these troll types allows us to establish a connection between troll types and the political bias of the news media they mention. Table 1 shows a summary of the troll role distribution, the total number of tweets per role, as well as examples of troll usernames and tweets.

4.2 Media Bias/Fact Check

We use data from Media Bias/Fact Check (MBFC)[2] to label news media sites. MBFC divides news media into the following bias categories: Extreme-Left, Left, Center-Left, Center, Center-Right, Right, and Extreme-Right. We reduce the granularity to three categories by grouping Extreme-Left and Left as LEFT, Extreme-Right and Right as RIGHT, and Center-Left, Center-Right, and Center as CENTER.

[1] http://github.com/fivethirtyeight/russian-troll-tweets

[2] http://mediabiasfactcheck.com

Bias	Count	Example
LEFT	341	www.cnn.com
RIGHT	619	www.foxnews.com
CENTER	372	www.apnews.com

Table 2: Summary statistics about the Media Bias/Fact Check (MBFC) dataset.

Table 2 shows some basic statistics about the resulting media dataset. Similarly to the IRA dataset, the distribution is right-heavy.

5 Experiments and Evaluation

5.1 Experimental Setup

For each user in the IRA dataset, we extracted all the links in their tweets, we expanded them recursively if they were shortened, we extracted the domain of the link, and we checked whether it could be found in the MBFC dataset. By grouping these relationships by media, we constructed the sets of users C_m that mention a given medium $m \in M$.

The U2H graph consists of 108 410 nodes and 443 121 edges, while the U2M graph has 591 793 nodes and 832 844 edges. We ran node2vec on each graph to extract 128-dimensional vectors for each node. We used these vectors as features for the fully supervised and for the distant-supervision scenarios. For Label Propagation, we used an empirical threshold for edge materialization $\tau = 0.55$, to obtain a reasonably sparse similarity graph.

We used two evaluation measures: accuracy, and macro-averaged F1 (the harmonic average of precision and recall). In the supervised scenario, we performed 5-fold cross-validation. In the distant-supervision scenario, we propagated labels from the media to the users. Therefore, in the latter case the user labels were only used for evaluation.

5.2 Evaluation Results

Table 3 shows the evaluation results. Each line of the table represents a different combination of features, models, or techniques. As mentioned in Section 3, the symbol '$\|$' denotes a single model trained on the concatenation of the features, while the symbol '$\oplus$' denotes an averaging of individual models trained on each feature separately. The tags 'LP1' and 'LP2' denote the two label propagation versions, by mention and by similarity, respectively.

We can see that accuracy and macro-averaged F1 are strongly correlated and yield very consistent rankings for the different models. Thus, henceforth we will focus our discussion on accuracy.

We can see in Table 3 that it is possible to predict the roles of the troll users by using distant supervision with relatively high accuracy. Indeed, the results for T2 are lower compared to their T1 counterparts by only 10 and 20 points absolute in terms of accuracy and F1, respectively. This is impressive considering that the models for T2 have no access to labels for troll users.

Looking at individual features, for both T1 and T2, the embeddings from U2M outperform those from U2H and from BERT. One possible reason is that the U2M graph is larger, and thus contains more information. It is also possible that the social circle of a troll user is more indicative than the hashtags they used. Finally, the textual content on Twitter is quite noisy, and thus the BERT embeddings perform slightly worse when used alone.

All our models with a single type of embedding easily outperform the model of Kim et al. (2019). The difference is even larger when combining the embeddings, be it by concatenating the embedding vectors or by training separate models and then combining the posteriors of their predictions.

By concatenating the U2M and the U2H embeddings (U2H $\|$ U2M), we fully leverage the hashtags and the mention representations in the latent space, thus achieving accuracy of 88.7 for T1 and 78.0 for T2, which is slightly better than when training separate models and then averaging their posteriors (U2H $\oplus$ U2M): 88.3 for T1 and 77.9 for T2. Adding BERT embeddings to the combination yields further improvements, and follows a similar trend, where feature concatenation works better, yielding 89.2 accuracy for T1 and 78.2 for T2 (compared to 89.0 accuracy for T1 and 78.0 for T2 for U2H $\oplus$ U2M $\oplus$ BERT).

Adding label propagation yields further improvements, both for LP1 and for LP2, with the latter being slightly superior: 89.6 vs. 89.3 accuracy for T1, and 78.5 vs. 78.3 for T2.

Overall, our methodology achieves sizable improvements over previous work, reaching an accuracy of 89.6 vs. 84.0 of Kim et al. (2019) in the fully supervised case. Moreover, it achieves 78.5 accuracy in the distant supervised case, which is only 11 points behind the result for T1, and is about 10 points above the majority class baseline.

Method	Full Supervision (T1)		Distant Supervision (T2)	
	Accuracy	Macro F1	Accuracy	Macro F1
Baseline (majority class)	68.7	27.1	68.7	27.1
Kim et al. (2019)	84.0	75.0	N/A	N/A
BERT	86.9	83.1	75.1	60.5
U2H	87.1	83.2	76.3	60.9
U2M	88.1	83.9	77.3	62.4
U2H $\oplus$ U2M	88.3	84.1	77.9	64.1
U2H $\|$ U2M	88.7	84.4	78.0	64.6
U2H $\oplus$ U2M $\oplus$ BERT	89.0	84.4	78.0	65.0
U2H $\|$ U2M $\|$ BERT	89.2	84.7	78.2	65.1
U2H $\|$ U2M $\|$ BERT + LP1	89.3	84.7	78.3	65.1
U2H $\|$ U2M $\|$ BERT + LP2	89.6	84.9	78.5	65.7

Table 3: Predicting the role of the troll users using full vs. distant supervision.

6 Discussion

6.1 Ablation Study

We performed different experiments with the hyper-parameters of the graph embeddings. With smaller dimensionality (i.e., using 16 dimensions instead of 128), we noticed 2–3 points of absolute decrease in accuracy across the board.

Moreover, we found that using all of the data for learning the embeddings was better than focusing only on users that we target in this study, namely *left*, *right*, and *news feed*, i.e., using the rest of the data adds additional context to the embedding space, and makes the target labels more contextually distinguishable. Similarly, we observe 5–6 points of absolute drop in accuracy when training our embeddings on tweets by trolls labeled as *left*, *right*, and *news feed*.

6.2 Comparison to Full Supervision

Next, we compared to the work of Kim et al. (2019), who had a fully supervised learning scenario, based on Tarde's Actor-Network Theory. They paid more attention to the content of the tweet by applying a text-distance metric in order to capture the semantic distance between two sequences. In contrast, we focus on critical elements of information that are salient in Twitter: *hashtags* and *user mentions*. By building a connection between users, hashtags, and user mentions, we effectively filtered out the noise and we focused only on the most sensitive type of context, thus automatically capturing features from this network via graph embeddings.

Method	Accuracy	Macro F1
Baseline (majority)	46.5	21.1
BERT	61.8	60.4
U2H	61.6	60.0
U2M	62.7	61.4
U2H $\oplus$ U2M	63.5	61.8
U2H $\|$ U2M	63.8	61.9
U2H $\oplus$ U2M $\oplus$ BERT	63.7	61.8
U2H $\|$ U2M $\|$ BERT	64.0	62.2

Table 4: Leveraging user embeddings to predict the bias of the media cited by troll users.

6.3 Reverse Classification: Media from Trolls

Table 4 shows an experiment in distant supervision for reverse classification, where we trained a model on the IRA dataset with the troll labels, and then we applied that model to the representation of the media in the MBFC dataset, where each medium is represented as the average of the embeddings of the users who cited that medium. We can see that we could improve over the baseline by 20 points absolute in terms of accuracy and by 41 in terms absolute in terms of macro-averaged F1.

We can see in Table 4 that the relative ordering in terms or performance for the different models is consistent with that for the experiments in the previous section. This suggests that the relationship between trolls and media goes both ways, and thus we can use labels for media as a way to label users, and we can also use labels for troll users as a way to label media.

7 Conclusion and Future Work

We have proposed a novel approach to analyze the behavior patterns of political trolls according to their political leaning (*left* vs. *news feed* vs. *right*) using features from social media, i.e., from Twitter. We experimented with two scenarios: (*i*) supervised learning, where labels for trolls are provided, and (*ii*) distant supervision, where such labels are not available, and we rely on more common labels for news outlets cited by the trolls. Technically, we leveraged the community structure and the text of the messages in the online social network of trolls represented as a graph, from which we extracted several types of representations, i.e., embeddings, for the trolls. Our experiments on the "IRA Russian Troll" dataset have shown improvements over the state-of-the-art in the supervised scenario, while providing a compelling case for the distant-supervision scenario, which has not been explored before.[3]

In future work, we plan to apply our methodology to other political events such as *Brexit* as well as to other election campaigns around the world, in connection to which large-scale troll campaigns have been revealed. We further plan experiments with other graph embedding methods, and with other social media. Finally, the relationship between media bias and troll's political role that we have highlighted in this paper is extremely interesting. We have shown how to use it to go from the media-space to the user-space and vice-versa, but so far we have just scratched the surface in terms of understanding of the process that generated these data and its possible applications.

Acknowledgments

This research is part of the Tanbih project,[4] which aims to limit the effect of "fake news", propaganda and media bias by making users aware of what they are reading. The project is developed in collaboration between the Qatar Computing Research Institute, HBKU and the MIT Computer Science and Artificial Intelligence Laboratory.

Gianmarco De Francisci Morales acknowledges support from Intesa Sanpaolo Innovation Center. The funder had no role in the study design, in the data collection and analysis, in the decision to publish, or in the preparation of the manuscript.

[3] Our data and code are available at http://github.com/amatanasov/conll_political_trolls

[4] http://tanbih.qcri.org/

References

Kayode Sakariyah Adewole, Nor Badrul Anuar, Amirrudin Kamsin, Kasturi Dewi Varathan, and Syed Abdul Razak. 2017. Malicious accounts: Dark of the social networks. *Journal of Network and Computer Applications*, 79:41–67.

Abdullah Almaatouq, Erez Shmueli, Mariam Nouh, Ahmad Alabdulkareem, Vivek K Singh, Mansour Alsaleh, Abdulrahman Alarifi, Anas Alfaris, et al. 2016. If it looks like a spammer and behaves like a spammer, it must be a spammer: analysis and detection of microblogging spam accounts. *International Journal of Information Security*, 15(5):475–491.

Amy Binns. 2012. DON'T FEED THE TROLLS! Managing troublemakers in magazines' online communities. *Journalism Practice*, 6(4):547–562.

Zhan Bu, Zhengyou Xia, and Jiandong Wang. 2013. A sock puppet detection algorithm on virtual spaces. *Know.-Based Syst.*, 37:366–377.

Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637.

Kevin R. Canini, Bongwon Suh, and Peter L. Pirolli. 2011. Finding credible information sources in social networks based on content and social structure. In *Proceedings of the IEEE Conference on Privacy, Security, Risk, and Trust, and the IEEE Conference on Social Computing*, SocialCom/PASSAT '11, pages 1–8, Boston, MA, USA.

Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on Twitter. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 675–684, Hyderabad, India.

Cheng Chen, Kui Wu, Venkatesh Srinivasan, and Xudong Zhang. 2013. Battling the internet water army: Detection of hidden paid posters. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 116–120, Niagara Falls, ON, Canada.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *Proceedings of the IEEE Conference on Privacy, Security, Risk and Trust and of the IEEE Conference on Social Computing*, PASSAT/SocialCom '12, pages 71–80, Amsterdam, Netherlands.

Kirsti K Cole. 2015. "It's like she's eager to be verbally abused": Twitter, trolls, and (en) gendering disciplinary rhetoric. *Feminist Media Studies*, 15(2):356–358.

Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2015.

Fame for sale: efficient detection of fake Twitter followers. *Decision Support Systems*, 80:56–71.

Kareem Darwish, Dimitar Alexandrov, Preslav Nakov, and Yelena Mejova. 2017. Seminar users in the Arabic Twitter sphere. In *Proceedings of the 9th International Conference on Social Informatics*, SocInfo '17, pages 91–108, Oxford, UK.

Kushal Dave, Steve Lawrence, and David M Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th International World Wide Web conference*, WWW '03, pages 519–528, Budapest, Hungary.

Michela Del Vicario, Alessandro Bessi, Fabiana Zollo, Fabio Petroni, Antonio Scala, Guido Caldarelli, H Eugene Stanley, and Walter Quattrociocchi. 2016. The spreading of misinformation online. *Proceedings of the National Academy of Sciences*, 113(3):554–559.

Chrysanthos Dellarocas. 2006. Strategic manipulation of internet opinion forums: Implications for consumers and firms. *Management Science*, 52(10):1577–1593.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '2019, pages 4171–4186, Minneapolis, MN, USA.

Ibrahim Elbeltagi and Gomaa Agag. 2016. E-retailing ethics and its impact on customer satisfaction and repurchase intention: A cultural and commitment-trust theory perspective. *Internet Research*, 26(1):288–310.

Michael Fire, Dima Kagan, Aviad Elyashar, and Yuval Elovici. 2014. Friend or foe? Fake profile identification in online social networks. *Social Network Analysis and Mining*, 4(1):1–23.

Patxi Galán-García, José Gaviria de la Puerta, Carlos Laorden Gómez, Igor Santos, and Pablo García Bringas. 2016. Supervised machine learning for the detection of troll profiles in Twitter social network: Application to a real case of cyberbullying. *Logic Journal of the IGPL*, 24(1):42–53.

Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2016. Quantifying controversy in social media. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, WSDM '16, pages 33–42, San Francisco, CA, USA.

Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2017. The effect of collective attention on controversial debates on social media. In *Proceedings of the 9th International ACM Web Science Conference*, WebSci '17, pages 43–52, Troy, NY, USA.

Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2018. Political discourse on social media: Echo chambers, gatekeepers, and the price of bipartisanship. In *Proceedings of the International World Wide Web Conference*, WWW '18, pages 913–922, Lyon, France.

Pepa Gencheva, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017. A context-aware approach for detecting worth-checking claims in political debates. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 267–276, Varna, Bulgaria.

Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, San Francisco, CA, USA.

Hong-Youl Ha, Joby John, J. Denise John, and Yongkyun Chung. 2016. Temporal effects of information from social networks on online behavior: The role of cognitive and affective trust. *Internet Research*, 26(1):213–235.

Momchil Hardalov, Ivan Koychev, and Preslav Nakov. 2016. In search of credible news. In *Proceedings of the 17th International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, AIMSA '16, pages 172–180, Varna, Bulgaria.

Naeemul Hassan, Chengkai Li, and Mark Tremayne. 2015. Detecting check-worthy factual claims in presidential debates. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, CIKM '15, pages 1835–1838, Melbourne, Australia.

Li-An Ho, Tsung-Hsien Kuo, and Binshan Lin. 2012. How social identification and trust influence organizational online knowledge sharing. *Internet Research*, 22(1):4–28.

Meng-Hsiang Hsu, Li-Wen Chuang, and Cheng-Se Hsu. 2014. Understanding online shopping intention: the roles of four types of trust and their antecedents. *Internet Research*, 24(3):332–352.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, Seattle, WA, USA.

Israa Jaradat, Pepa Gencheva, Alberto Barrón-Cedeño, Lluís Màrquez, and Preslav Nakov. 2018. ClaimRank: Detecting check-worthy claims in Arabic and English. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, New Orleans, LA, USA.

Andreas M Kaplan and Michael Haenlein. 2010. Users of the world, unite! The challenges and opportunities of social media. *Business horizons*, 53(1):59–68.

Georgi Karadzhov, Pepa Gencheva, Preslav Nakov, and Ivan Koychev. 2017a. We built a fake news & clickbait filter: What happened next will blow your mind! In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 334–343, Varna, Bulgaria.

Georgi Karadzhov, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, and Ivan Koychev. 2017b. Fully automated fact checking using external sources. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 344–353, Varna, Bulgaria.

Dongwoo Kim, Timothy Graham, Zimin Wan, and Marian-Andrei Rizoiu. 2019. Tracking the digital traces of Russian trolls: Distinguishing the roles and strategy of trolls on Twitter. *CoRR*, abs/1901.05228.

Edward C. S. Ku. 2012. Beyond price, how does trust encourage online group's buying intention? *Internet Research*, 22(5):569–590.

Srijan Kumar, Justin Cheng, Jure Leskovec, and V.S. Subrahmanian. 2017. An army of me: Sockpuppets in online discussion communities. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 857–866, Perth, Australia.

Srijan Kumar, Francesca Spezzano, and VS Subrahmanian. 2014. Accurately detecting trolls in slashdot zoo via decluttering. In *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Network Analysis and Mining*, ASONAM '14, pages 188–195, Beijing, China.

David M. J. Lazer, Matthew A. Baum, Yochai Benkler, Adam J. Berinsky, Kelly M. Greenhill, Filippo Menczer, Miriam J. Metzger, Brendan Nyhan, Gordon Pennycook, David Rothschild, Michael Schudson, Steven A. Sloman, Cass R. Sunstein, Emily A. Thorson, Duncan J. Watts, and Jonathan L. Zittrain. 2018. The science of fake news. *Science*, 359(6380):1094–1096.

Wenbin Li, Ning Zhong, and Chunnian Liu. 2006. Combining multiple email filters based on multivariate statistical analysis. In *Foundations of Intelligent Systems*, pages 729–738. Springer.

Xiaodong Li, Xinshuai Guo, Chuang Wang, and Shengliang Zhang. 2016. Do buyers express their true assessment? Antecedents and consequences of customer praise feedback behaviour on Taobao. *Internet Research*, 26(5):1112–1133.

Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2018. Attributed social network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 30(12):2257–2270.

Darren L Linvill and Patrick L Warren. 2018. Troll factories: The internet research agency and state-sponsored agenda building. *Resource Centre on Media Freedom in Europe*.

Clare Llewellyn, Laura Cram, Robin L. Hill, and Adrian Favero. 2018. For whom the bell trolls: Shifting troll behaviour in the Twitter Brexit debate. *JCMS: Journal of Common Market Studies*, 57(5):1148–1164.

Michal Lukasik, Trevor Cohn, and Kalina Bontcheva. 2015. Point process modelling of rumour dynamics in social media. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 518–523, Beijing, China.

Jing Ma, Wei Gao, Prasenjit Mitra, Sejeong Kwon, Bernard J. Jansen, Kam-Fai Wong, and Meeyoung Cha. 2016. Detecting rumors from microblogs with recurrent neural networks. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI '16, pages 3818–3824, New York, NY, USA.

Michael McCord and M. Chuah. 2011. Spam detection on Twitter using traditional classifiers. In *Proceedings of the 8th International Conference on Autonomic and Trusted Computing*, ATC '11, pages 175–186, Banff, Canada.

Todor Mihaylov, Georgi Georgiev, and Preslav Nakov. 2015a. Finding opinion manipulation trolls in news community forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, CoNLL '15, pages 310–314, Beijing, China.

Todor Mihaylov, Ivan Koychev, Georgi Georgiev, and Preslav Nakov. 2015b. Exposing paid opinion manipulation trolls. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, RANLP '15, pages 443–450, Hissar, Bulgaria.

Todor Mihaylov, Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Georgi Georgiev, and Ivan Koychev. 2018. The dark side of news community forums: Opinion manipulation trolls. *Internet Research*, 28(5):1292–1312.

Todor Mihaylov and Preslav Nakov. 2016. Hunting for troll comments in news community forums. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 399–405, Berlin, Germany.

Tsvetomila Mihaylova, Georgi Karadzhov, Pepa Atanasova, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, SemEval '19, pages 860–869, Minneapolis, MN, USA.

Tsvetomila Mihaylova, Preslav Nakov, Lluís Màrquez, Alberto Barrón-Cedeño, Mitra Mohtarami, Georgi Karadjov, and James Glass. 2018. Fact checking in community forums. In *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI '18, pages 879–886, New Orleans, LA, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, Lake Tahoe, NV, USA.

Tanushree Mitra, Graham P. Wright, and Eric Gilbert. 2017. A parsimonious language model of social media credibility across disparate events. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, pages 126–145, Portland, OR, USA.

Mitra Mohtarami, Ramy Baly, James Glass, Preslav Nakov, Lluís Màrquez, and Alessandro Moschitti. 2018. Automatic stance detection using end-to-end memory networks. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 767–776, New Orleans, LA, USA.

Michael J Moore, Tadashi Nakano, Akihiro Enomoto, and Tatsuya Suda. 2012. Anonymity and roles associated with aggressive posts in an online forum. *Computers in Human Behavior*, 28(3):861–867.

Meredith Ringel Morris, Scott Counts, Asta Roseway, Aaron Hoff, and Julia Schwarz. 2012. Tweeting is believing?: Understanding microblog credibility perceptions. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 441–450, Seattle, WA, USA.

Preslav Nakov, Tsvetomila Mihaylova, Lluís Màrquez, Yashkumar Shiroya, and Ivan Koychev. 2017. Do not trust the trolls: Predicting credibility in community question answering forums. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, RANLP '17, pages 551–560, Varna, Bulgaria.

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-party deep network representation. *Network*, 11(9):12.

Kashyap Popat, Subhabrata Mukherjee, Jannik Strötgen, and Gerhard Weikum. 2017. Where the truth lies: Explaining the credibility of emerging claims on the web and social media. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17, pages 1003–1012, Perth, Australia.

Jacob Ratkiewicz, Michael Conover, Mark Meiss, Bruno Gonçalves, Snehal Patil, Alessandro Flammini, and Filippo Menczer. 2011. Truthy: Mapping the spread of astroturf in microblog streams. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 249–252, Hyderabad, India.

Carlos Ruiz, David Domingo, Josep Lluís Micó, Javier Díaz-Noci, Koldo Meso, and Pere Masip. 2011. Public sphere 2.0? The democratic qualities of citizen debates in online newspapers. *The International Journal of Press/Politics*, 16(4):463–487.

Jari Salo and Heikki Karjaluoto. 2007. A conceptual model of trust in the online environment. *Online Information Review*, 31(5):604–621.

Chun-Wei Seah, Hai Leong Chieu, Kian Ming Adam Chai, Loo-Nin Teow, and Lee Wei Yeong. 2015. Troll detection by domain-adapting sentiment analysis. In *Proceedings of the 18th International Conference on Information Fusion*, FUSION '15, pages 792–799, Washington, DC, USA.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Pnina Shachaf and Noriko Hara. 2010. Beyond vandalism: Wikipedia trolls. *Journal of Information Science*, 36(3):357–370.

Thamar Solorio, Ragib Hasan, and Mainul Mizan. 2014. Sockpuppet detection in Wikipedia: A corpus of real-world deceptive writing for linking identities. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, LREC '14, pages 1355–1358, Reykjavik, Iceland.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, Florence, Italy.

Scott Thacker and Mark D Griffiths. 2012. An exploratory study of trolling in online video gaming. *International Journal of Cyber Behavior, Psychology and Learning (IJCBPL)*, 2(4):17–33.

Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science*, 359(6380):1146–1151.

Mengzhou Zhuang, Geng Cui, and Ling Peng. 2018. Manufactured opinions: The effect of manipulating online product reviews. *Journal of Business Research*, 87:24 – 35.

Arkaitz Zubiaga, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Peter Tolmie. 2016. Analysing how people orient to and spread rumours in social media by looking at conversational threads. *PLoS ONE*, 11(3):1–29.

Towards a Unified End-to-End Approach for Fully Unsupervised Cross-lingual Sentiment Analysis

Yanlin Feng and **Xiaojun Wan**
Wangxuan Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{fengyanlin,wanxiaojun}@pku.edu.cn

Abstract

Sentiment analysis in low-resource languages suffers from the lack of training data. Cross-lingual sentiment analysis (CLSA) aims to improve the performance on these languages by leveraging annotated data from other languages. Recent studies have shown that CLSA can be performed in a fully unsupervised manner, without exploiting either target language supervision or cross-lingual supervision. However, these methods rely heavily on unsupervised cross-lingual word embeddings (CLWE), which has been shown to have serious drawbacks on distant language pairs (e.g. English - Japanese). In this paper, we propose an end-to-end CLSA model by leveraging unlabeled data in multiple languages and multiple domains and eliminate the need for unsupervised CLWE. Our model applies to two CLSA settings: the traditional cross-lingual in-domain setting and the more challenging cross-lingual cross-domain setting. We empirically evaluate our approach on the multilingual multi-domain Amazon review dataset. Experimental results show that our model outperforms the baselines by a large margin despite its minimal resource requirement. [1]

1 Introduction

While English sentiment analysis has achieved great success with the help of large-scale annotated corpus, this is not the case for most of languages where only limited data is available. Cross-lingual sentiment analysis (CLSA) tackles this problem by adapting the sentiment resource in a source language to a poor-resource language (the target language).

Current state-of-the-art CLSA methods rely heavily on cross-lingual word embeddings (CLWE) to transfer sentiment information from the source language to the target language. CLWE encodes words from multiple languages in a common space, thus making it possible to share a classifier across languages. Recent studies have shown that CLWE can be obtained in an unsupervised way, i.e., without any cross-lingual resources (Zhang et al., 2017; Conneau et al., 2017; Artetxe et al., 2018). This motivates fully unsupervised CLSA approaches (Chen et al., 2018a) that do not rely on either target language supervision or cross-lingual supervision. These methods generally involve the following steps:

1. Train monolingual embeddings separately on multiple languages using monolingual unlabeled data.

2. Map the monolingual embeddings to a shared space using unsupervised CLWE methods, either adversarial training methods (Conneau et al., 2017) or non-adversarial methods (Artetxe et al., 2018; Xu et al., 2018).

3. Train a sentiment classifier using the annotated corpus in the source language.

However, it has been shown that the quality of unsupervised CLWE is highly sensitive to the choice of language pairs and the comparability of the monolingual data (Søgaard et al., 2018). Therefore, these methods fail when the source language and the target language have very different linguistic structures (e.g. English and Japanese) and require additional cross-lingual supervision (e.g. a small seed dictionary or shared identical strings) in such cases (Chen et al., 2018a).

In this paper, we propose a unified end-to-end framework to perform unsupervised CLSA, bypassing the complex multi-step process and the drawbacks of unsupervised CLWE methods. Instead of mapping monolingual embeddings to a

[1]The source code is available at https://github.com/Evan-Feng/UXSenti

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 1035–1044
Hong Kong, China, November 3-4, 2019. ©2019 Association for Computational Linguistics

shared continuous space, we propose to bridge the language gap by multilingual multi-domain language modeling (i.e., we model the probabilities of sentences from multiple language-domain pairs). The language modeling objective is jointly trained with a classification objective in an end-to-end fashion using the unlabeled data in multiple language-domain pairs and labeled data in a source language-domain pair. Our model applies to two CLSA settings: the traditional cross-lingual in-domain setting and the more challenging cross-lingual cross-domain setting.

The rationale for using unlabeled data in multiple domains is that there may not be a domain shared by all languages in low-resource scenarios. If we want to perform CLSA on two languages that only have resources in two different domains, it is natural to bridge the language gap with another language that have resources on both domains. Even in the case where resources in a specific domain are available for all languages, which is a common assumption made by most CLSA approaches, we show that exploiting unlabeled data in other domains significantly improves performance.

Our contributions are as follows:

1. We propose a unified end-to-end framework to perform CLSA. Our approach is fully unsupervised and does not rely on any form of cross-lingual supervision (even shared identical strings) or target language supervision.

2. We show that cross-lingual language modeling based methods are able to outperform CLWE based methods in the unsupervised setting.

3. Our model can be easily generalized to different CLSA settings. Experiments on the multilingual multi-domain Amazon review dataset show that our method achieves state of the art in both the cross-lingual in-domain setting and the cross-lingual cross-domain setting despite its minimal resource requirement.

2 Related Work

Cross-lingual Sentiment Analysis The most related topic to our work is cross-lingual sentiment analysis. Some CLSA methods rely on machine translation systems (Wan, 2009; Demirtas and Pechenizkiy, 2013; Xiao and Guo, 2012; Zhou et al., 2016a) to provide cross-lingual supervision, making themselves implicitly dependant on large-scale parallel corpus which may not be available for low-resource languages. Wan (2009) apply the co-training algorithm to translated data while other researchers have proposed multi-view learning (Xiao and Guo, 2012).

Another line of CLSA research bridges the language gap using CLWE, which saves the efforts of training a machine translation system thus requires less cross-lingual resources. Some work has proposed to map pretrained monolingual embeddings to a shared space (Barnes et al., 2018) to obtain CLWE while others proposed jointly learning CLWE and a sentiment classifier, allowing the embeddings to encode sentiment information (Zhou et al., 2016b; Xu and Wan, 2017).

Very recently, unsupervised CLSA methods that do not require either cross-lingual supervision or target language supervision have been proposed (Chen et al., 2018b,a). Chen et al. (2018a) transfer sentiment information from multiple source languages by jointly learning language invariant and language specific features. Yet, these unsupervised CLSA methods rely on unsupervised CLWE which builds on the assumption that pretrained monolingual embeddings can be properly aligned. This assumption, however, is not true in low-resource scenarios (Søgaard et al., 2018).

It is worth pointing out that the language-adversarial training model of (Chen et al., 2018b) is able to perform unsupervised CLSA without CLWE. The proposed model consists of a feature extractor, a sentiment classifier and a language discriminator. The feature extractor is trained to fool the discriminator so that the extracted features are language invariant. However, its performance is significantly lower than the variant that uses pretrained CLWE.

While traditional CLSA methods assume that data in both languages is within the same domain (e.g. English hotel reviews for training and Chinese hotel review for testing, we refer to this setting as "cross-lingual in-domain sentiment analysis"), the more challenging cross-lingual cross-domain setting has also been explored. Ziser and Reichart (2018) extend pivot-based monolingual domain adaption methods to the cross-lingual setting. However, their method is not unsupervised and requires expensive cross-lingual resources.

Cross-lingual Language Modeling Our work is also related to cross-lingual language modeling, which is a topic that has been explored by researchers very recently. Lample and Conneau (2019), pretrain a language model with a joint vocabulary on the concatenation of multiple large-scale monolingual corpora and finetune it on labeled data. However, this approach exploits cross-lingual supervision provided by shared sub-word units, which has been shown to improve performance (Lample et al., 2018), and it remains a challenge to efficiently perform cross-lingual transfer without exploiting shared identical strings. In this work, we treat identical words from different languages as different words and thus eliminate any form of cross-lingual supervision.

Wada and Iwata (2018) proposed a similar cross-lingual language modeling architecture for unsupervised word translation. They show that it outperforms mapping based approaches (Artetxe et al., 2018; Lample et al., 2017), but only when a small amount of monolingual data is used. The difference between their model and ours is that we adopt different parameter sharing strategies and consider the correlation between multiple domains.

3 Cross-lingual In-Domain Sentiment Analysis

3.1 Overview

In this section we describe our cross-lingual in-domain sentiment analysis model (CLIDSA). It assumes the training data and test data come from different languages but are within the same domain (e.g. English hotel reviews as training data and Chinese hotel reviews as test data), which is the most common setting of previous CLSA approaches.

Although we use the same set of labeled data as previous CLSA approaches, we adopt a different strategy for utilizing the unlabeled data. Suppose there is a set of languages $\mathcal{L}$ and a set of domains $\mathcal{D}$. Let $\mathcal{P} \subseteq \mathcal{L} \times \mathcal{D}$ denote a set of language-domain pairs. For each language-domain pair $(l, d) \in \mathcal{P}$, we have a set of unlabeled reviews $\mathcal{C}^{l,d}_{mono}$. We also have a annotated sentiment corpus $\mathcal{C}^{l_s,d_s}_{senti}$ in a source language-domain pair (l_s, d_s). Our goal is to predict the sentiment polarity of the examples in a target language-domain pair (l_t, d_t) (note that $d_s = d_t$ in the cross-lingual in-domain setting).

In Section 5 we compare two CLIDSA variants. CLIDSA$_{full}$ exploits unlabeled data from all possible language-domain pairs, i.e., we set $\mathcal{P} = \mathcal{L} \times \mathcal{D}$. However, since most previous CLSA methods do not use multi-domain or multilingual unlabeled data, we create a variant CLIDSA$_{min}$ that requires minimal resources by setting $\mathcal{P} = \{(l_s, d_s), (l_t, d_t)\}$.

A natural way to utilize unlabeled data is to perform the language modeling task. Our CLIDSA model consists of multiple language models for mutiple language-domain pairs, with some of their parameters shared across languages or across domains. It also includes a classifier component which takes the hidden states (produced by the LSTM language model) as input features and predicts the sentiment polarity. We also adopt a language discriminator to force the features to be language invariant. The overall architecture of our model is illustrated in Figure 1. We detail each component of our CLIDSA model in the following subsections.

3.2 Multilingual Multi-Domain Language Modeling

Language modeling is the most critical part in our model since it acts as a language invariant feature extractor. Intuitively, if we share the LSTM layers of language models across languages, these layers are likely to process sentences from different languages in the same space, thus inducing language invariant features. In this subsection we detail our parameter sharing strategies for modeling sentences from multiple language-domain pairs.

Following previous work, we compute the probability of a sentence x by modeling the probability of a word w_k given the previous words:

$$p(x) = \prod_{k=1}^{|x|} p(w_k \mid w_1, \ldots, w_k - 1) \qquad (1)$$

For sentences in a certain language-domain pair (l, d), the probabilities are computed using a two-layer LSTM language model, which includes an embedding layer, two LSTM layers and a linear decoding layer. We first pass the input words through the embedding layer of language l which is parameterized by θ^l_{emb}. Then we forward the word embeddings to a LSTM layer parameterized by θ_{lstm1}, which is shared across all languages and all domains, generating a sequence of intermediate

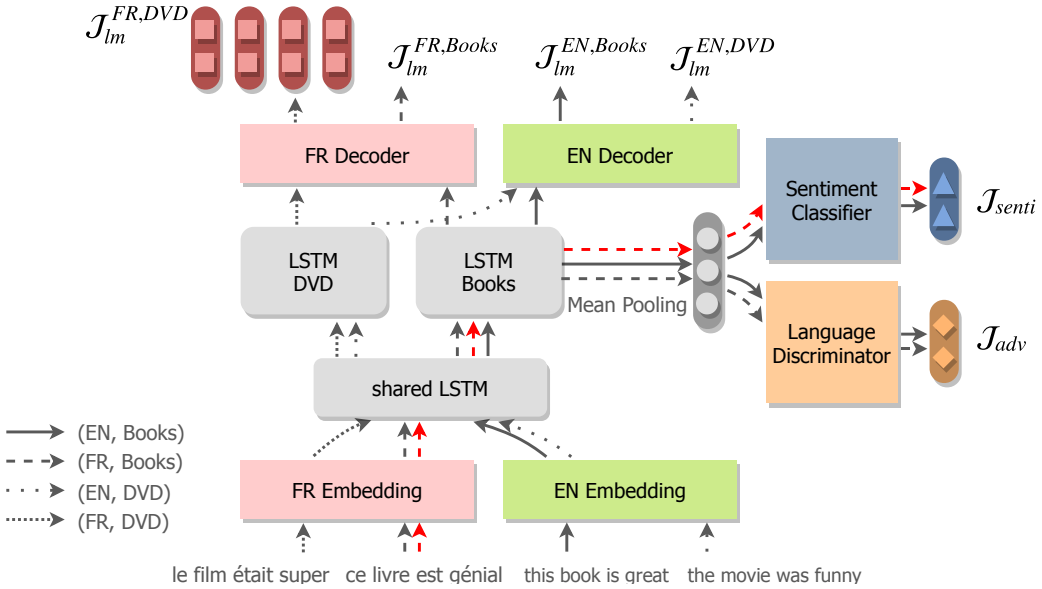


Figure 1: Illustration of the CLIDSA model. In this example, $\mathcal{L} = \{\text{EN}, \text{FR}\}$, $\mathcal{D} = \{\text{Books}, \text{DVD}\}$, $\mathcal{P} = \mathcal{L} \times \mathcal{D}$, $l_s = \text{EN}$, $l_t = \text{FR}$ and $d_s = d_t = \text{Books}$. We visualize the forward pass of input sentences from different language-domain pairs. The path shown in red only occurs at test time.

hidden states:

$$h_k = \text{LSTM}(h_{k-1}, \vec{w}_k; \theta_{lstm1}) \tag{2}$$

where $\vec{w}_k$ denotes the embedding of word w_k. These hidden states are then passed through the second LSTM layer which is domain specific but language invariant, generating a sequence of final hidden states:

$$z_k = \text{LSTM}(z_{k-1}, h_k; \theta^d_{lstm2}) \tag{3}$$

where the second LSTM layer of domain d is parameterized by θ^d_{lstm2}. The final hidden states $(z_1, z_2, \ldots, z_{|x|})$ thus can be considered as language invariant features for cross-lingual classification.

For the purpose of language modeling, we adopt a language-specific linear decoding layer to transform the final hidden states into probability distributions for next word prediction. The decoding layer of language l is parameterized by θ^l_{dec} and is shared across domains.

The intuition of adopting a domain-specific LSTM layer is that the distribution of sentences varies across domains. For example, given the first three words "I love this", the next word is most likely to be "book" in a book review dataset or "movie" in a movie review dataset. While it is possible to address this issue by using domain-specific linear decoding layers, we find that sharing the decoders across domains substantially reduces the total number of parameters thus provides regularization when only limited resources are available (see Section 5.5 for the ablation study). Sharing the decoders further enables the weight tying technique (Inan et al., 2016) to tie the decoder weight with the embedding layer.

For language-domain pair (l, d), the language modeling objective is written as follows:

$$\mathcal{J}^{l,d}_{lm}(\theta^l_{emb}, \theta_{lstm1}, \theta^d_{lstm2}, \theta^l_{dec}) = \mathbb{E}_{x \sim \mathcal{C}^{l,d}_{mono}} \left[-\frac{1}{|x|} \sum_{k=1}^{|x|} \log p(w_k \mid w_1, \ldots, w_{k-1}) \right] \tag{4}$$

where the sentence likelihood is normalized by the sentence length and $x \sim \mathcal{C}^{l,d}_{mono}$ indicates that x is sampled from the unlabeled text in $\mathcal{C}^{l,d}_{mono}$.

3.3 Sentiment Classifier

We adopt a simple linear classifier that takes the averaged final hidden states $\frac{1}{|x|}\sum_{k=1}^{|x|} z_k$ as input features and outputs the probabilities of different labels. The classification objective can be written as:

$$\mathcal{J}_{senti}(\theta^{l_s}_{emb}, \theta_{lstm1}, \theta^{d_s}_{lstm2}, \theta_{clf}) = \mathbb{E}_{(x,y) \sim \mathcal{C}^{l_s,d_s}_{senti}} \big[-\log p(y \mid x) \big] \tag{5}$$

where $(x, y) \sim \mathcal{C}^{l_s,d_s}_{senti}$ indicates that the sentence x and its label y are sampled from the source sen-

timent corpus and θ_{clf} denotes the parameters of the linear classifier.

The classification objective is jointly minimized with the language modeling objective, allowing sentiment-specific supervision signals to back-propagate through the model so that it can learn to extract useful features for sentiment prediction.

3.4 Language Adversarial Training

To further force the features used for sentiment classification to be language invariant, we adopt the language adversarial training technique (Chen et al., 2018b). A language discriminator is trained to predict the language ID given the features by minimizing the cross entropy loss, while the LSTM network is trained to fool the discriminator by maximizing the loss:

$$\mathcal{J}_{adv}(\theta_{emb}, \theta_{lstm1}, \theta^{d_s}_{lstm2}, \theta_{dis}) = \mathbb{E}_{(x,l)}[-\log p(l \mid x)] \quad (6)$$

where $\theta_{emb} = \theta^1_{emb} \oplus \cdots \oplus \theta^{|\mathcal{L}|}_{emb}$ denotes the parameters of all the embedding layers and θ_{dis} denotes the parameters of the language discriminator. The sentence x and the language id l are sampled from all the unlabeled data in domain $d_s = d_t$. We do not employ language adversarial training on the features of other domain since we only perform classification in a single domain.

3.5 The Full Objective Function

Putting all the components together, the final objective function is thus:

$$\mathcal{J}_{full}(\theta_{emb}, \theta_{lstm}, \theta_{dec}, \theta_{clf}, \theta_{dis}) = \sum_{(l,d)\in\mathcal{P}} \mathcal{J}^{l,d}_{lm} + \alpha\mathcal{J}_{senti} - \beta\mathcal{J}_{adv} \quad (7)$$

where $\theta_{lstm} = \theta_{lstm1} \oplus \theta^1_{lstm2} \oplus \cdots \oplus \theta^{|\mathcal{D}|}_{lstm2}$ denotes the parameters of all the LSTM layers, $\theta_{dec} = \theta^1_{dec} \oplus \cdots \oplus \theta^{|\mathcal{L}|}_{dec}$ denotes the parameters of all the decoding layers, α and β are the hyper-parameters controlling the importance of the classification objective and the language adversarial training objective. Parameters θ_{dis} are trained to maximize this objective function while the others are trained to minimize it:

$$\hat{\theta}_{dis} = \arg\max_{\theta_{dis}} \mathcal{J}_{full} \quad (8)$$

$$(\hat{\theta}_{emb}, \hat{\theta}_{lstm}, \hat{\theta}_{dec}, \hat{\theta}_{clf}) = \arg\min_{\theta_{emb},\theta_{lstm},\theta_{dec},\theta_{clf}} \mathcal{J}_{full} \quad (9)$$

4 Cross-lingual Cross-Domain Sentiment Analysis

In this section we focus on a more challenging CLSA setting, where the training data and test data are from different languages and different domain (e.g. English hotel reviews as training data, Chinese book reviews as test data). We show that the CLIDSA model can be applied to this setting with only slight modification.

Following previous notations, we denote the source language-domain pair as (l_s, d_s) and the target language pair as (l_t, d_t) with $l_s \neq l_t$ and $d_s \neq d_t$. Ziser and Reichart (2018) rely on expensive resources to perform cross-lingual cross-domain transfer, including unlabeled data from (l_s, d_s) and (l_t, d_t), CLWE and machine translation. In this work, we also use the unlabeled data from (l_s, d_s) and (l_t, d_t). However, instead of relying on CLWE and machine translation, we propose to leverage the unlabeled data in a "pivot" pair (l_s, d_t) to bridge the language-domain gap. This is reasonable since source languages are those with rich resources and we do not use additional annotation. Formally, we have a set of unlabeled reviews for each language-domain pair in $\mathcal{P} = \{(l_s, d_s), (l_s, d_t), (l_t, d_t)\}$ and a set of labeled reviews from (l_s, d_s).

In the CLIDSA model, inputs from (l_t, d_t) do not go through the source-domain LSTM layer (parameterized by $\theta^{l_s}_{lstm2}$), thus can not be forwarded to the sentiment classifier for sentiment prediction. Nevertheless, we now show that we can directly apply the CLIDSA model to this setting by slightly altering the forward pass of (l_t, d_t). Figure 2 illustrates our CLCDSA model for cross-lingual cross-domain sentiment analysis. The architecture of CLCDSA is identical to CLIDSA (i.e. parameterized by the same set of parameters), but the data forwarding process is slightly different. The key idea is simple: instead of viewing the sentiment classifier as a linear classifier that takes the domain-specific final hidden states $z_1, z_2, \ldots, z_{|x|}$ as input features, we consider the source-domain LSTM layer and the linear classifier together as a "LSTM+Linear" classifier (parameterized by $\theta^{d_s}_{lstm2} \oplus \theta_{clf}$) that takes the domain invariant and language invariant hidden states $h_1, h_2, \ldots, h_{|x|}$ as input features. From

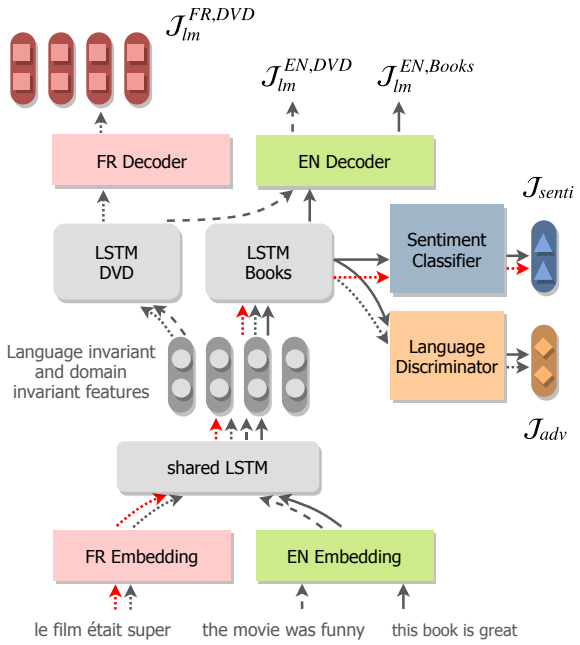


Figure 2: Illustration of the CLCDSA model. In this example, $(l_s, d_s) = (\text{EN}, \text{Books})$ and $(l_t, d_t) = (\text{FR}, \text{DVD})$. The architecture of CLCDSA is identical to CLIDSA, but the data forwarding process is different. We visualize the forward pass of input sentences from different language-domain pairs. The path shown in red only occurs at test time.

this point of view, we can pass the first-layer hidden states to the source-domain LSTM layer and the sentiment classifier to obtain the sentiment prediction at test time.

At training time, the first-layer hidden states generated from a target sentence are forwarded to the source-domain LSTM layer ($\theta_{lstm2}^{d_s}$) and the language discriminator (θ_{dis}) to compute the adversarial loss. The LSTM layers are trained to fool the language discriminator so that it cannot distinguish the examples in (l_s, d_s) from those in (l_t, d_t). We jointly optimize three language modeling objectives for each language-domain pair, the adversarial objective, and a sentiment classification objective for (l_s, d_s).

	EN	DE	FR	JA
Books	50000	165470	32870	169780
DVD	30000	91516	9358	68326
Music	25220	60392	15940	55892

Table 1: Number of unlabeled examples in the Amazon dataset.

5 Experiments

5.1 Datasets

We evaluate our model on the multilingual multi-domain Amazon review dataset (Prettenhofer and Stein, 2010) which contains product reviews in four languages (English, French, German, Japanese) and three domains (Books, DVD, Music). For each language-domain pair, there are 2000 examples for training and 2000 examples for testing. The statistics of unlabeled data is summarized in Table 1. For cross-lingual in-domain sentiment analysis, we use English as the source language and the others as target languages, resulting in nine tasks in total. For cross-lingual cross-domain sentiment analysis, we follow the setting in (Ziser and Reichart, 2018) and use English as the source language, French and German as target languages, and consider all the domain combinations, resulting in twelve tasks in total. Note that we would also want to evaluate our model on some low resource languages. However, since there isn't an public benchmark for such languages, we leave it to future work.

5.2 Implementation Details

Most of the hyperparamters are set empirically without tuning. For language modeling, we adopt the AWD-LSTM language model (Merity et al., 2017) with 1150 hidden units and a weight dropout rate of 0.5. We refer readers to (Merity et al., 2017) for a more detailed description. The sentiment classifier is a linear classifier with a dropout rate of 0.6. The language discriminator is a three-layer MLP with 400 hidden units.

As the only exceptions, hyperparameters α and β are tuned on the target development set following standard CLSA practice. We set (α, β) to $(0.01, 0.1)$ for CLIDSA$_{full}$ and CLCDSA, $(0.01, 0.03)$ for CLIDSA$_{min}$ in all tasks and do not perform any task-specific tuning.

The Adam optimizer (Kingma and Ba, 2014) with a base learning rate of 0.003 and $\beta_1 = 0.7$ is used for training. In each iteration, we sample a batch from every language-domain pairs in $\mathcal{P}$ to compute the language modeling loss and discriminator loss. Then we sample a batch from the source annotated corpus to compute the sentiment classification loss. All the parameters are jointly updated using the Gradient Reversal Layer (Ganin et al., 2016) and standard backpropagation. We

	EN-DE			EN-FR			EN-JA		
	Books	DVD	Music	Books	DVD	Music	Books	DVD	Music
Methods with cross-lingual supervsion									
CL-SCL†‡	79.50	76.92	77.79	78.49	78.80	77.92	73.09	71.07	75.11
BiDRL†‡	84.14	84.05	84.67	84.39	83.60	82.52	73.15	76.78	78.77
UMM†	81.65	81.27	81.32	80.27	80.27	79.41	71.23	72.55	75.38
CLDFA†‡	83.95	83.14	79.02	83.37	82.56	83.31	77.36	80.52	76.46
Methods without cross-lingual supervision									
MAN-MoE	82.40	78.80	77.15	81.10	84.25	80.90	62.78	69.10	72.60
MWE	76.10	76.80	74.70	76.35	78.70	71.60	-	-	-
CLIDSA$_{min}$	86.55	80.35	83.50	86.65	85.40	84.30	75.90	71.45	71.40
CLIDSA$_{full}$	**86.65**	**84.60**	**85.05**	**87.20**	**87.95**	**87.15**	**79.35**	**81.90**	**84.05**

Table 2: Test accuracy of different CLSA methods on the Amazon review dataset in the cross-lingual in-domain setting. The highest score on each task is shown in **bold**. The second highest score is underlined. '-'indicates that MUSE fails to align the EN and JA embeddings so MWE's predictions are random. Methods that require cross-lingual resources are marked as †. Methods that require machine translation are marked as ‡.

run 50000 iterations for CLIDSA and 30000 iterations for CLCDSA without early stopping.

5.3 Baselines

We compare our model to the following CLSA baselines, including methods that require cross-lingual resources (either in the form of machine translation or parallel data), methods that rely on unsupervised CLWE, and a few variants of our proposed model. **PBLM-BE** is a cross-lingual cross-domain model, **MWE** applies to both settings, while others are cross-lingual in-domain methods.

CL-SCL Prettenhofer and Stein (2010) map the bag-of-word representations to a cross-lingual space via structural correspondence learning.

BiDRL Zhou et al. (2016b) learn bilingual document representation for CLSA. The authors translate each document into both languages and enforce a bilingual constraint between the original document and the translated version.

UMM Xu and Wan (2017) jointly learn multilingual word embeddings and a sentiment classifier using parallel corpora of multiple language pairs. Languages that do not have direct parallel corpus are bridged via a third pivot language.

CLDFA Xu and Yang (2017) propose cross-lingual distillation using translated reviews.

MAN-MoE Chen et al. (2018a) propose the state-of-the-art unsupervised CLSA model that learns language invariant features and language specific features. It relies on unsupervised CLWE for cross-lingual transfer. Unlike other CLSA approaches, it transfers the sentiment information from multiple source languages.

MWE This is a variant of our proposed model that relies on unsupervised CLWE instead of language modeling. We map all target language embeddings to the English space using the MUSE library (Conneau et al., 2017) and use them to initialize the embedding layers. We train the sentiment classifier using the labeled data in the source language-domain pair and directly apply it to the test data. The same architecture is used but we only optimize the classification objective.

PBLM-BE Ziser and Reichart (2018) extend existing pivot-based domain adaption approaches to the cross-lingual settings using CLWE and machine translation.

5.4 Results and Analysis

Cross-lingual In-Domain Results Table 2 presents the performance of different CLSA methods on various cross-lingual in-domain tasks. Our proposed model achieves new state of the art on all nine tasks. Even in the restricted setting where only minimal resources are used (no cross-lingual resources, no pretrained embeddings, no multilingual multi-domain unlabeled data), CLIDSA$_{min}$ outperforms the strongest baseline on four out of nine tasks, validating the efficacy of our proposed model. Exploiting multilingual multi-domain unlabeled data leads to an average improvement of $+4.27\%$ across all tasks. We

	EN-DE						EN-FR					
	D-B	M-B	B-D	M-D	B-M	D-M	D-B	M-B	B-D	M-D	B-M	D-M
PMLM-BE†	78.7	78.6	80.6	79.2	81.7	78.5	81.1	74.7	76.3	75.0	75.1	76.8
MWE	76.3	72.8	74.7	72.5	74.2	76.0	74.8	72.4	76.0	74.2	72.5	74.3
CLCDSA	**85.4**	**81.7**	**79.3**	**81.0**	**83.4**	**81.7**	**86.2**	**81.8**	**84.3**	**82.8**	**83.7**	**85.0**

Table 3: Test accuracy of different CLSA methods on the Amazon review dataset in the cross-lingual cross-domain setting. The highest score on each task is shown in **bold**. Methods that require cross-lingual resources are marked as †. The abbreviations {B, D, M} stand for {Books, DVD, Music}.

	EN-DE	EN-FR	EN-JA
CLIDSA$_{full}$	**84.6**	**88.0**	**81.9**
- decoder sharing	83.0	87.0	78.9
- LSTM-1 sharing	82.4	87.1	81.4
- discriminator	82.6	87.4	81.6
- joint training	81.2	86.7	79.9

Table 4: Ablation results in the cross-lingual in-domain setting. English is used as the source language and DVD is used as the source/target domain. The highest score for each language pair is shown in **bold**.

also find that it is most beneficial to sentiment analysis on distant language pairs, with an average improvement of $+8.85\%$ on EN-JA.

Among methods that do not require cross-lingual resources, CLWE based methods are lower than the proposed cross-lingual language modeling based methods. This is interesting because it has been shown in previous work (Wada and Iwata, 2018) that cross-lingual language modeling does not perform well on the word translation task when sufficient monolingual data is available. Nevertheless, we demonstrate that this is not the case for cross-lingual sentiment analysis.

Cross-lingual Cross-Domain Results Table 3 shows the results of various cross-lingual cross-domain tasks. MWE suffers greatly from domain discrepancy compared to the in-domain results. Nevertheless, our model outperforms all baselines on all tasks, with an average improvement of $+5\%$ across all tasks.

5.5 Ablation Study

We perform an ablation study to investigate the contribution of individual components. The results are summarized in Table 4. We first create a variant that does not share the decoding layers across domain, and another one that does not share the first LSTM layer across domain. Disabling parameter sharing hurts the performance most on EN-JA (-1.75%). We also observed that the performance gap is much more significant when less training data is used (not shown here).

Surprisingly, removing the language discriminator does not lead to significant performance drop, which indicates that the language modeling alone is able to produce language invariant features. Intuitively, parameter sharing would force the LSTM layers to process sentences from different languages in the same space, thus inducing cross-lingual feature representation. Note that we also try removing the language modeling objective and rely on language adversarial training to provide cross-lingual features, but find that the resulting performance is rather poor.

Finally, we explore a different training strategy where the sentiment classifier is not jointly trained with the other components. Instead, we use the labeled data to train the classifier only after we have trained the other components on the unlabeled data. We observe that the resulting performance drop is due to underfitting, i.e., the extracted features do not encode enough information for sentiment prediction. This highlights the importance of end-to-end training.

6 Conclusion and Future Work

In this work we present an end-to-end approach for cross-lingual sentiment analysis. Our method is fully unsupervised thus does not rely on any cross-lingual supervision and target language supervision. We rely on language modeling to provide language invariant feature representations. We propose two model variants, one for cross-lingual in-domain transfer and the other for cross-lingual cross-domain transfer. Both models achieve state of the art on the Amazon review dataset. Experimental results also show that exploiting multilingual multi-domain unlabeled data greatly benefits CLSA on distant language pairs.

There are several straight-forward extensions

of our model: cross-lingual in-domain sentiment analysis with multiple source languages, cross-lingual cross-domain sentiment analysis with multiple target languages, etc. We leave the exploration of these extensions to future work.

Acknowledgment

This work was supported by National Natural Science Foundation of China (61772036) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. 2018. Bilingual sentiment embeddings: Joint projection of sentiment across languages. *arXiv preprint arXiv:1805.09016*.

Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. 2018a. Zero-resource multilingual model transfer: Learning what to share. *arXiv preprint arXiv:1810.03552*.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018b. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Erkin Demirtas and Mykola Pechenizkiy. 2013. Cross-lingual polarity detection with machine translation. In *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining*, page 9. ACM.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. *arXiv preprint arXiv:1804.07755*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and Optimizing LSTM Language Models. *arXiv preprint arXiv:1708.02182*.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1118–1127.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. *arXiv preprint arXiv:1805.03620*.

Takashi Wada and Tomoharu Iwata. 2018. Unsupervised cross-lingual word embedding by multilingual neural language models. *arXiv preprint arXiv:1809.02306*.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-volume 1*, pages 235–243. Association for Computational Linguistics.

Min Xiao and Yuhong Guo. 2012. Multi-view adaboost for multilingual subjectivity analysis. *Proceedings of COLING 2012*, pages 2851–2866.

Kui Xu and Xiaojun Wan. 2017. Towards a universal sentiment classifier in multiple languages. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 511–520.

Ruochen Xu and Yiming Yang. 2017. Cross-lingual distillation for text classification. *arXiv preprint arXiv:1705.02073*.

Ruochen Xu, Yiming Yang, Naoki Otani, and Yuexin Wu. 2018. Unsupervised cross-lingual transfer of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2465–2474.

Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1959–1970.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016a. Attention-based lstm network for cross-lingual sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 247–256.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016b. Cross-lingual sentiment classification with bilingual document representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1403–1412.

Yftah Ziser and Roi Reichart. 2018. Deep pivot-based modeling for cross-language cross-domain transfer with minimal guidance. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 238–249.

Author Index

Abdou, Mostafa, 87
Abend, Omri, 174, 291
Abercrombie, Gavin, 249
Aharoni, Roee, 196
Ahmad, Wasi Uddin, 372
Ahmed, Sajawel, 871
Allauzen, Cyril, 121
Alshaikh, Rana, 131
Amac, Mustafa Sercan, 441
Andreoli, Jean-Marc, 900
Andueza Rodriguez, Maria, 110
Angeli, Gabor, 393
Aralikatte, Rahul, 87
Atanasov, Atanas, 1023
Azab, Mahmoud, 99

Bagchi, Saurabh, 708
Bai, Ziwei, 593
Baldridge, Jason, 55, 528
Batista-Navarro, Riza, 249
Beaufays, Françoise, 121
Beekhuizen, Barend, 77
Benton, Adrian, 574
Berant, Jonathan, 452
Besacier, Laurent, 339
Betke, Margrit, 504
Biecek, Przemyslaw, 624
Boschee, Elizabeth, 515
Bouraoui, Zied, 131
Broscheit, Samuel, 677

Cai, Jingjing, 789
Chaganty, Arun Tejasvi, 393
Chang, Angel X., 393
Chang, Kai-Wei, 152, 372
Chaturvedi, Snigdha, 645
Chawla, Kushal, 833
Chen, Guanyi, 383
Chen, Haochen, 152
Chen, Jianshu, 316, 696
Chen, Jiayu, 593
Chen, Kunlong, 961
Chen, Mingqing, 121
Chen, Muhao, 152
Cheng, Renhong, 970
Cheng, Xingyi, 961
Chevrot, Jean-Pierre, 339
Chhaya, Niyati, 833
Choshen, Leshem, 291
Chu, Wei, 961
Coheur, Luísa, 582
Collinson, Matthew, 383
Cotterell, Ryan, 140
Cui, Yiming, 737

Dagan, Ido, 196
Dai, Qiong, 163
De Francisci Morales, Gianmarco, 1023
de la Torre, Antonio, 538
Deng, Jia, 99
Deutsch, Daniel, 482
Diallo, Aïssatou, 910
Dieter, Justin, 393
Dong, Li, 515
Driller, Christine, 871
Dyer, Chris, 227
Dymetman, Marc, 900

Elliott, Desmond, 87
Erdem, Aykut, 441
Erdem, Erkut, 441
Eskenazi, Maxine, 582

Feng, Yanlin, 1035
Feng, Yukun, 920
Fischer, Andreas, 890
Freedman, Marjorie, 656
Fung, Pascale, 271
Fürnkranz, Johannes, 910

Galstyan, Aram, 666
Gambäck, Björn, 940
Gangavarapu, Tushaar, 1012
Gao, Jianwei, 613
Gao, Li, 281
Gao, Meng, 862
Gao, Yanjun, 404
Garcia-Olano, Diego, 528
Gillick, Daniel, 528

Goldberg, Yoav, 463
Gonen, Hila, 463
Gorman, Kyle, 140
Guo, Honglei, 970
Guo, Lei, 504
Guo, Siwen, 992
Gupta, Arshit, 798
Gurevych, Iryna, 493

Han, Rujun, 666
Hanselowski, Andreas, 493
Havard, William N., 339
He, Shizhu, 718
He, Su, 737
He, Xiaodong, 747
Heinzerling, Benjamin, 216
Hessel, Jack, 419
Höhn, Sviatlana, 992
Hollenstein, Nora, 538
Holzenberger, Nils, 44
Hovy, Eduard, 349
Hsu, I-Hung, 666
Hu, Guoping, 737
Hu, J. Edward, 44
Hu, Junfeng, 961
Hu, Mengting, 970
Hu, Yue, 281
Huang, Heyan, 260
Huang, Hsin-Ping, 686
Huang, Jianghua, 163
Huang, Jing, 747
Huang, Kevin, 747
Huang, Weipeng, 961
Huang, Xiao, 515
Huang, Zuying, 822
Hupkes, Dieuwke, 1

Ie, Eugene, 528
Ilharco, Gabriel, 55

Jain, Rishabh, 929
Jamatia, Anupam, 940
Jha, Rahul, 757
Ji, Yuze, 613
Jian, Ping, 260
Jiang, Lei, 163
Jumelet, Jaap, 1

Kamath, Sowmya, 1012
Kamigaito, Hidetaka, 920
Karmaker Santu, Shubhra Kanti, 778
Kasai, Jungo, 304
Kementchedjhieva, Yova, 463
Kim, Kunho, 757
Klami, Arto, 634
Knaebel, René, 768
Kocher, Noémien, 890
Kocoń, Jan, 980
Kojima, Noriyuki, 99
Korhonen, Anna, 33, 216
Koshorek, Omri, 452
Krishnan, Gokul S, 1012
Kulkarni, Sayali, 528
Kumar, Anuj, 728
Kumar, Vishwajeet, 812

Lacroix, Mathieu, 238
Lagus, Jarkko, 634
Lai, Yi-An, 798
Lamm, Matthew, 87
Langer, Nicolas, 538
Lansing, Larry, 528
Lazaridis, Alexandros, 890
Le Roux, Joseph, 238
Lei, Kai, 862
Li, Junyi Jessy, 686
Li, Lei, 822
Li, Ruizhe, 383
Li, Xiao, 383
Li, Yuan-Fang, 812
Li, Zile, 493
Lin, Chenghua, 383
Lin, Youfang, 613
Lin, Ziqi, 1002
Linzen, Tal, 66
Litvak, Marina, 822
Liu, Cao, 718
Liu, Fenglin, 862
Liu, Kang, 718
Liu, Qianchu, 33
Liu, Siyi, 504
Liu, Ting, 737
Liu, Wei, 822
Liu, Yuanxin, 862
Lo, Chi-kiu, 206

Ma, Wentao, 737
Ma, Xuezhe, 372
Madhyastha, Pranava, 929
Madotto, Andrea, 271
Mahgoub, Ashraf, 708
Manning, Christopher D., 843
Mansour, Riham, 708
Marin, Alex, 757
Markowska, Magdalena, 140

Mathews, Rajiv, 121
Matthews, Austin, 227
May, Jonathan, 656
Mayhew, Stephen, 645
Mays, Kate, 504
McAllester, David, 696
McCallum, Andrew, 574
McCarthy, Arya D., 140
McCarthy, Diana, 33
Mehler, Alexander, 871
Merlo, Paola, 110
M'hamdi, Meryem, 656
Michaelis, Laura, 362
Mihalcea, Rada, 99
Miłkowski, Piotr, 980
Moeller, Sarah, 362
Moon, Seungwhan, 728
Mota, Pedro, 582
Mulcaire, Phoebe, 304
Murauer, Benjamin, 951
Musat, Claudiu, 890
Myer, Kriti, 574
Mysore, Sheshera, 574

Naik, Aakanksha, 349
Nakov, Preslav, 1023
Nanni, Federico, 249
Nayak, Tapas, 603
Neishi, Masato, 328
Neubig, Graham, 227
Ng, Hwee Tou, 603
Ni, Wancheng, 1002
Nie, Zaiqing, 718
Nikolaus, Mitja, 87
Ning, Qiang, 550
Noseworthy, Michael, 430

Okumura, Manabu, 920

Pachzelt, Adrian, 871
Palmer, Martha, 362
Pang, Bo, 419
Pappu, Aneesh, 843
Park, Daehyung, 430
Parshakova, Tetiana, 900
Passonneau, Rebecca J., 404
Patil, Pallavi, 574
Paul, Rohan, 430
Pei, Zhengqi, 881
Peng, Haoruo, 550
Peng, Huailiang, 163
Peng, Nanyun, 372, 515, 666
Ponzetto, Simone Paolo, 249
Post, Matt, 44
Prange, Jakob, 174
Prasad, Grusha, 66
Presta, Alessandro, 528

Rabinovich, Ella, 77
Radinsky, Kira, 186
Ramakrishnan, Ganesh, 812
Ravichander, Abhilasha, 349
Rehbein, Ines, 472
Reichart, Roi, 216
Riley, Michael, 121
Rose, Carolyn, 349
Rosin, Guy D., 186
Roth, Dan, 482, 550, 645, 696
Roy, Nicholas, 430
Roy, Subhro, 430
Rozen, Ohad, 196
Rozenknop, Antoine, 238

Sakuma, Jin, 22
Sarkar, Anoop, 12
Saxena, Rohun, 843
Schneider, Nathan, 174
Schockaert, Steven, 131
Schommer, Christoph, 992
Schulz, Claudia, 493
Schumann, Raphael, 472
Scuito, Christian, 890
See, Abigail, 843
Shah, Pararth, 728
Shahin, Youssef, 708
Shao, Nan, 737
Shi, Xuewen, 260
Shindo, Hiroyuki, 563
Shwartz, Vered, 196
Silfverberg, Miikka, 140
Simard, Michel, 206
Singh, Abhinav, 44
Singh, Arpit, 574
Sinkkonen, Janne, 634
Skiena, Steven, 152
Smith, Noah A., 304
Soricut, Radu, 419
Specht, Günther, 951
Srikumar, Vivek, 452
Srinivasan, Balaji Vasan, 833
Stab, Christian, 493
Stanislawek, Tomasz, 624
Stanovsky, Gabriel, 452
Stede, Manfred, 768

Stent, Amanda, 574
Stevenson, Suzanne, 77
Stober, Sebastian, 768
Stoeckel, Manuel, 871
Stowe, Kevin, 362
Strube, Michael, 216
Su, Zhong, 970
Subba, Rajen, 728
Sun, Chen, 404
Sun, Kai, 316, 696
Sun, Zhewei, 881
Suresh, Ananda Theertha, 121
Swamy, Steve Durairaj, 940

Takamura, Hiroya, 920
Tang, Yi-Kun, 260
Tang, Yun, 747
Tarantino, Lorenzo, 890
Tian, Yingtao, 152
Tong, Chaodong, 163
Tsai, Chen-Tse, 645
Tschuggnall, Michael, 951

Upadhyay, Shyam, 482

Van Durme, Benjamin, 44
van Schijndel, Marten, 66
Vanetik, Natalia, 822
Veeramachaneni, Kalyan, 778
Vulić, Ivan, 33, 216
Vylomova, Ekaterina, 140

Wan, Huaiyu, 613
Wan, Xiaojun, 1035
Wang, Hai, 316, 696
Wang, Ji, 789
Wang, Shijin, 737
Wang, Taifeng, 961
Wang, Tian, 393
Wang, Wenguan, 260
Wang, Xiaojie, 593
Watson, Julia, 77
Wei, Xiangpeng, 281
Weischedel, Ralph, 666
Wijaya, Derry Tanti, 504
Williams, Kyle, 757
Winata, Genta Indra, 271
Wójcicka, Alicja, 624
Wong, Adeline, 121
Wróblewska, Anna, 624
Wu, Chien-Sheng, 271

Xing, Luxi, 281
Xu, Jianjun, 789
Xu, Weidi, 961
Xu, Yang, 881

Yagcioglu, Semih, 441
Yamada, Ikuya, 563
Yang, Mu, 666
Yerukola, Akhila, 843
Yoshinaga, Naoki, 22, 328
Yu, Dian, 316, 696
Yu, Dong, 316, 696
Yuan, Caixia, 593

Zala, Ronak, 574
Zaniolo, Carlo, 152
Zaśko-Zielińska, Monika, 980
Zhai, Chengxiang, 778
Zhang, Ce, 538
Zhang, Haidong, 1002
Zhang, Haoyu, 789
Zhang, Wei-Nan, 737
Zhang, Yi, 798
Zhang, Yuan, 55
Zhang, Zhisong, 372
Zhao, Jun, 718
Zhao, Meijing, 1002
Zhao, Shiwan, 970
Zhou, Bowen, 747
Zhou, Yichu, 452
Zhu, Yi, 216
Zhu, Zhenhai, 419
Zhu, Zhenqi, 12
Ziembicki, Daniel, 624
Zitouni, Imed, 757
Zopf, Markus, 910
Zuidema, Willem, 1