CCNLG 2019

4th Workshop on Computational Creativity in Language Generation

Proceedings of the Workshop

November 1, 2019
Tokyo, Japan

Sponsored by The University of Antwerp

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 978-1-950737-67-3

Introduction

Welcome to the 4th Workshop on Computational Creativity in Language Generation, a workshop held in conjunction with INLG 2019, the International Conference on Natural Language Generation, in Tokyo, Japan.

Discussions at CC-NLG will cover the distinct approaches of CC and NLG brought about by their respective focuses; research in computational creativity has tended to deal less with technical shifts, directed instead at cognition, aesthetics, and novelty; whilst NLG research has tended to focus on the technical and theoretical challenges of topics like readability. However, in recent years this distinction has become far less defined. NLG research deals actively with concepts of style, variation, poetics, and narrative, whilst creative researchers are developing robust implementations. This change can be seen in dialogue systems, where the usability of an interface relies on it handling out-of-domain or spontaneous user input. Creative methodologies are garnering fundamental and applicable returns.

—The Organizers

Table of Contents

Toward Automated Quest Generation in Text-Adventure Games

Prithviraj Ammanabrolu, William Broniec, Alex Mueller,
Jeremy Paul, and Mark O. Riedl
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA, USA
{raj.ammanabrolu,wbroniec3,alexmueller,jeremypaul,riedl}@gatech.edu

Abstract

Interactive fictions, or text-adventures, are games in which a player interacts with a world entirely through textual descriptions and text actions. Text-adventure games are typically structured as puzzles or quests wherein the player must execute certain actions in a certain order to succeed. In this paper, we consider the problem of procedurally generating a quest, defined as a series of actions required to progress towards a goal, in a text-adventure game. Quest generation in text environments is challenging because they must be semantically coherent. We present and evaluate two quest generation techniques: (1) a Markov chains, and (2) a neural generative model. We specifically look at generating quests about cooking and train our models on recipe data. We evaluate our techniques with human participant studies looking at perceived creativity and coherence.

1 Introduction

Natural language can be used to express creativity in the form of narrative. Prior research has shown that narrative is used in everything from environmental understanding (Bruner, 1991) to developing language (Johnston, 2008). Given this wide ranging impact, using narrative in language to help us understand human perceptions of creativity and what it takes to replicate this through computational models is natural. Text-adventure games or interactive fiction, in which a player interacts with a world entirely through text, provide us with a platform on which to explore these ideas on creativity in language. These games are usually structured as puzzles or quests in which a player must complete a sequence of actions in order to succeed. Text games allow us to factorize the problem of creative language generation and focus on developing more fine-grained, data-driven models.

Automated generation of text-adventure games can broadly be split into two considerations: (1) the structure of the world, including the layout of rooms, textual description of rooms, objects, and non-player characters; and (2) the quest, consisting of the partial ordering of activities that the player must engage in to make progress toward the end of the game. In this work, we focus on methods of automatically generating such a quest and how it can be used to better understand narrative intelligence, specifically looking at perceived creativity and coherence. Quest generation requires narrative intelligence as a quest must maintain coherence throughout and progress towards a goal. Maintaining quest coherence also means following the constraints of the given game world. The quest has to fit within the confines of the world in terms of both genre and given affordances—e.g. using magic in a fantasy world. This is further complicated in the case of a text-adventure as a consequence of all interactions being in natural language—the potential output space is combinatorial in size. Because the player "sees" and "acts" entirely through text, any quest generation system must also take into account the lack of visual information and generate sufficiently descriptive text accordingly.

There are multiple variables that could potentially affect a player's perception of creativity in a text-adventure game such as the vocabulary used, the structure of the world, stylistic variations in writing, etc. We use the TextWorld framework (Côté et al., 2018) which lets us generate text-adventure game worlds based on a grammar. It lets us fix variables concerned with game world and logic generation and focus only on the generation of quests within this world. We use this framework's "home" theme—providing us with a textual simulation of a house—and restrict the types of quests that can be generated to those involving the completion of a cooking recipe. We then attempt to learn how to generate a quest to complete a recipe—as well as how to create the recipe itself—using a large scale knowledge base of recipes. In these quests, players are provided with a list of ingredients and their locations, and they have to navigate the environment to find and prepare those ingredients to complete the given recipe. For example, given a recipe to make peanut butter cookies the quest would first tell the player to find eggs, peanut butter, flour, and baking soda. The player would then have to figure out that the first ingredient is in the fridge while the others are in the pantry and prepare each item accordingly. Generating this sort of quest requires knowledge of the ingredients, how they fit together, and how those ingredients interact with the environment.

The contribution of this work is thus twofold. We first detail a framework, and variations thereof, that can learn to generate creative quests in a text-adventure game. This framework includes two quest generation models using Markov chains as well as a neural language model. It also uses a semantically grounded knowledge graph to improve overall quest coherence. Our second contribution provides human subject evaluations that give us insight into how each variation of this framework affects human perception of creativity and coherence in such games.

2 Related Work

Although there has been much work recently on text-adventure gameplay (Bordes et al., 2010; He et al., 2016; Narasimhan et al., 2015; Fulda et al., 2017; Yang et al., 2018; Haroush et al., 2018; Côté et al., 2018; Tao et al., 2018; Ammanabrolu and Riedl, 2019a; Hausknecht et al., 2019a; Ammanabrolu and Riedl, 2019b; Hausknecht et al., 2019b), these works focus on creating agents that can play a given game as opposed to being able to automatically generate content for them.

Outside of this, there has been some work on learning to create content in the context of interactive narrative. These systems mainly work to overcome a significant bottleneck in the form of the human authoring required to create such works. Permar and Magerko (2013) present a method of generating cognitive scripts required for freeform activities in the form of pretend play. Specifically, they use interactive narrative—a form of pretend play that requires a high level of improvisation and creativity and uses cognitive scripts acquired from multiple experience sources. They take existing cognitive scripts and blend them in the vein of more traditional conceptual blending (Veale et al., 2000; Zook et al., 2011) to create new blended scripts. Closely related is Magerko et al. (2014) who present a Co-Creative Cognitive Architecture (CoCoA), detailing the set of components that support the design of co-creative agents in the context of interactive narrative. These methods all follow singular cognitive models that do not learn to generate content automatically.

Li et al. (2012) present Scheherazade, a system which learns a plot graph based on stories written by crowd sourcing the task of writing short stories through Amazon Mechanical Turk. This plot graph contains details relevant for the coherence of the story and includes: plot events, temporal precedence, and mutual exclusion relations. The generated narrative contains events that can be executed from this plot graph by both players and non-player characters. Guzdial et al. (2015) introduce Scheherazade-IF, a system that learns to generate choose-your-own-adventure style interactive fictions in which the player chooses from prescribed options. More recently, Martin et al. (2017) introduce a pipeline systems for improvisational storytelling agents capable of collaboratively creating stories. These agents first focus on creating a plot for the story and then expand that plot into natural language sentences. Urbanek et al. (2019) introduce Light, a dataset of crowdsourced text-adventure game dialogs focusing on giving collaborative agents the ability to generate contextually relevant dialog and emotes.

Giannatos et al. (2011) use genetic algorithms to create new story plot points for an existing game of interactive fiction using an encoding known as a precedence-constraint graph. This graph gives the system information regarding the ordering of events that must happen in the game in order to advance. They demonstrate the workings of their system by generating additional content for the popular interactive fiction game *Anchorhead*, and show that this can be integrated into the original game. This work, however, is offline and relies on existing interactive fiction games and having knowledge of the precedence-constraint graph for this existing game.

The Game Forge system (Hartsook et al., 2011) also uses genetic algorithms to generate a game

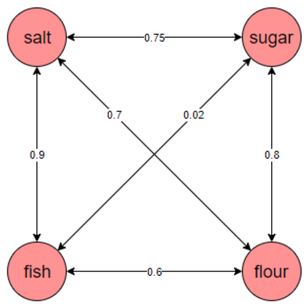


Figure 1: Example of ingredient connections.

world and plot line for related type of game, a computer role playing game (CRPG). This work focuses on generating layouts and plot structures to create novel game worlds through with a fitness function based on a transition graph that encodes pre-built game requirements. Tamari et al. (2019) focus on extracting action graphs for sequential decision making problems such as material science experiments and turn them into text-adventure games. Although these works use graph structures in order to constrain the generation of the game, we use these graph structures only to maintain coherence and focus on content creation.

Although there are works that attempt to automatically evaluate the creativity of the output of a generative process by computationally modeling potential human responses — such as with story telling (Purdy et al., 2018), etc. — we choose to rely on a human subject study based on the definition of creativity as presented in Boden (2007). Specifically we focus on the concepts of novelty and value, despite collecting data for other defined metrics as well. We use the definition of novelty stemming from the idea of p-creativity, i.e. a concept that is entirely new to a single agent – in this case a subject in our evaluation study. Value, as a component of computational creativity, however, is not defined concretely in Boden's work for a general domain. Our definition of value in the context of text-adventure games relies on accomplishment or achievement.

3 Content Generation

In this section, we present Markov chain and neural language model based models to generate content, i.e. recipes, for our quests. Content generation for a quest in a text-adventure game, in this case a recipe, can be thought of as being equivalent to generating a sequence of events in which prior elements affect the probability of subsequent events. Markov chains present a simplified and well studied method to generate such content. Neural language models, designed to predict an element of a sequence conditioned on a given number of prior elements, let us generate sequences of events with more prior context—i.e. in the absence of the Markov assumption.

3.1 Markov Chains

Our first quest generation model is based on the use of Markov chains. This generation process consists of two steps. We first learn a weighted *ingredient graph*, a Markov chain, from a large scale knowledge base of recipes and then probabilistically walk along this graph to generate the instructions for the recipe.

3.1.1 Ingredient Graph

Generating the recipe requires domain knowledge. For example, creating a recipe for peanut butter cookies requires an understanding that an ingredient like peanut butter fits well with eggs, flour, and sugar while something like fish does not. We represent this knowledge with an undirected graph of ingredients. Our ingredient graph is based off of recipes scraped from `allrecipes.com`. [1] The raw, uncleaned dataset included over 20,000 recipes with over 4000 unique ingredients. A list of ingredients was extracted from each recipe, and each of these lists was converted into a set of ingredient pairs (Fig. 2). In total, there were 118,116 unique ingredient pairings, and 73,088 of those pairings (62%) only occurred once. We reduced the number of distinct ingredients from 4460 to 1703 by merging items with the same base ingredient and by replacing name-brand items with a generic equivalent.

Each of the nodes within the graph represents a possible ingredient, and weighted connections between these nodes represent how well the ingredients go together. The weight of each edge is the total number of occurrences of that ingredient pair within the recipe corpus. The edge connecting eggs and white sugar would have a weight of 3774 while the edge between hot milk and orange juice would have a weight of 1. Ingredient pairings that do not occur within the recipe corpus did not have an edge within this network, and this helped

[1] `https://github.com/kbrohkahn/recipe-parser`

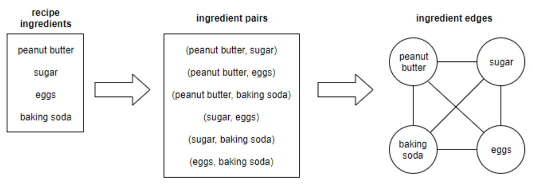


Figure 2: Ingredient extraction process.

prevent our model from generating completely incoherent recipe pairings (e.g. hot sauce and baby food). Take the graph in Fig. 1 as an example. In this complete graph, all of the ingredients go well with each other except for fish and sugar, which is indicated by the low weight connection between them. The weak connection between sugar and fish suggest that they would likely not go well together in a recipe.

3.1.2 Instruction Generation

With the ingredient graph created, we begin the process of instruction generation based on subgraph mining and prior generative methods based on probabilistic graph walks (Fleishman, 1978). We start by selecting an initial random ingredient 'x' weighted by its distribution in the graph.

$$p(x_1) = \frac{\sum_{i=1}^{k} w(v_i, x_1)}{\sum_{i=1}^{k} \sum_{j=1}^{k} w(v_i, v_j)} \tag{1}$$

We probabilistically select one of its neighbors based on the conditional frequency of the pair. Each iteration further computes conditional probabilities and selects them. We exclude all ingredients in which any bag of words token is contained by any other, ensuring that a variety of different ingredients are selected.

$$\alpha = \begin{cases} 0 & B_{x_i} \subseteq B_{x_{n+1}} \vee B_{x_{n+1}} \subseteq B_{x_i} \\ 1 & else \end{cases} \tag{2}$$

In Eq. 2, $\mathrm{B_{x_i}}$ refers to the 1-gram bag of words model.

However, just computing complete conditional probabilities would remove the chance for entirely new combinations to emerge. Therefore, we calculate just the partial probability of having shared ingredients with a bias designed to favor such combinations.

$$\beta = \left(\sum_{i=1}^{n} Shared(x_i, x_{n+1})\right)^2 \tag{3}$$

$$Shared(x_1, x_2) = \begin{cases} 1 & w(x_1, x_2) > 0 \\ 0 & else \end{cases} \tag{4}$$

This process repeated recursively to generate a recipe with the desired number of ingredients.

$$p(x_{n+1}) = \sum_{i=1}^{n} \alpha\beta \frac{w(x_{n+1}, v_j)}{\sum_{j=1}^{k} w(x_i, v_j)} \tag{5}$$

Finally, resultant combinations are referenced back against the original corpus to guarantee novelty in the result.

3.2 Neural Language Model

Here we use a neural language model to generate both the ingredients for a recipe and the steps of the ingredients as well. We use the same knowledge base as described in Sec. 3.1.1 and train two separate language models: one to generate the ingredients, and the other to generate the recipe given a set of ingredients.

The first language model uses a simple 4-layer LSTM to generate a sequence of ingredients, treating all the words in a single ingredient as a single token. For example, "peanut butter" would be considered a single token in this model. We train this model using the sets of ingredients found in each recipe for the entire recipe dataset, with each set ending with an <EOI> or End of Ingredients tag. Once trained, the model then generates a sequence of ingredients until the <EOI> is reached using the top-k sampling technique (Holtzman et al., 2019).

To generate the actual recipe, we use GPT-2 (Radford et al., 2019) and fine-tune their pretrained 345m parameter model on the recipe data. The data to fine-tune this model is designed to contain the recipe title, ingredients, and instructions in an unstructured text-form. Once this model has been fine-tuned, we use it to generate the title and instructions for the recipe conditioned on the ingredients generated by the first language model. The entire generated recipe consists of the ingredients, title, and instructions.

4 Quest Assembly

We now use the generated content, i.e. the recipe, to assemble a quest—grounding the generated in-

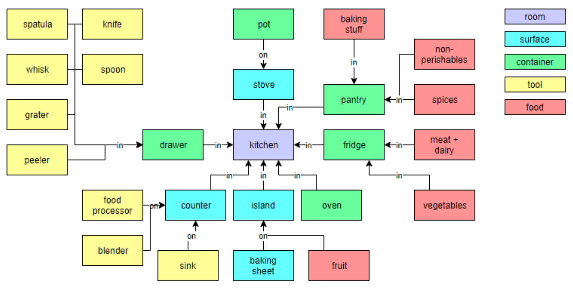


Figure 3: Object graph in the one room map.

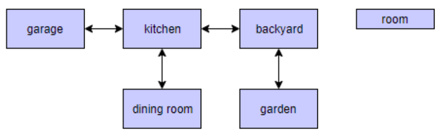


Figure 4: Room layout in the five room map.

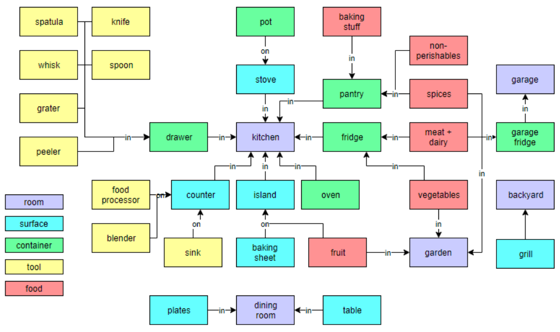


Figure 5: Object graph in the five room map.

gredients and instructions in the game world. This requires us to first determine the structure of the game world and the locations of objects within this world in addition to transforming the set of generated instructions into executable actions. We use two types of semantically grounded knowledge graphs to represent this information: the object and action graphs.

The object graph is used to determine the structure of the world and the most likely locations of objects within this world. For example, we could have information that says that vegetables must be stored in a refrigerator. If a recipe requires carrots, then the carrots would automatically be placed in a refrigerator at the start of the game. This graph is constructed by hand and is built to make the game world and resulting quest as coherent as possible.

We construct object graphs for two different room layouts. The first, the one room (1R) map, consists of a kitchen as well as the objects and actions that exist within it. The second map, the five room (5R) map, is an extension of the first map and contains four additional rooms.

The object graph for the 1R map as shown in Fig. 3 is largely inspired by the simple, prebuilt game provided within TextWorld (Côté et al., 2018). This object graph determines how and where objects are placed within the environment during game generation, and the action graph (Fig.6) dictates how generated instructions are transformed into executable actions in the game. The object graph was constructed logically: tools and utensils go in the drawer, meat and dairy belong in the refrigerator, and so on. Food item placements are deterministic and coherent. Vegetables always go in the refrigerator, and fruit always goes on the kitchen island. The action graph was also designed to prevent the player from conducting illogical actions.

The 5R map included a dining room, garage, backyard, and garden in addition to the kitchen (Fig. 4). The map (Fig. 5) is designed to maintain the same levels of coherency as the 1R map while allowing for more diverse gameplay, which could in turn lead to higher levels of perceived creativity. The additional rooms are selected based on their possible relationships to the domain of food and cooking, and each new room has its own unique objects that players can interact with. For example, the garage has an old refrigerator that can be used to store meat. These new rooms and objects also allow for dynamic food placement. Meat can be placed in one of two refrigerators, and fruits and vegetables can possibly be found in the garden. The existence of these new locations is not immediately clear to the player. The garage and backyard are additionally obscured by closed doors, adding to quest complexity. While the additional rooms and dynamic food placement allow for more diverse gameplay, they do not sacrifice coherency.

The action graph contains information regarding the affordances of the objects in the world and what kinds of objects are required to complete a given generated instruction. For example, if a generated instruction tells us to prepare vegetables, i.e. cut them, then this graph tells us that there must be a knife somewhere in this world. This graph is partially extracted from static cooking guides online using a mixture of OpenIE (Angeli

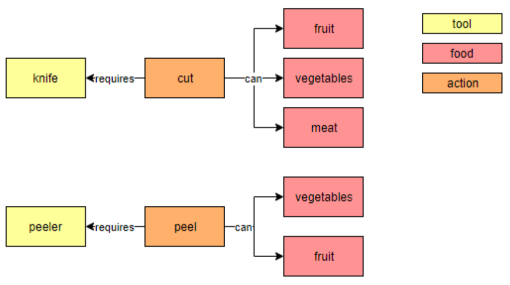


Figure 6: Example action graph for both maps.

et al., 2015) and hand-authored rules to account for the irregularities of cooking guides. An example of an action graph is given in Fig. 6. A player can peel fruit and vegetables, for example, but cannot peel a steak. There are also strict rules on what tools are required for certain actions. A player can only cut something if they have a knife and can only peel something with a peeler. While this restricts how players can interact with the environment, it ultimately reinforces game coherency.

We also note that when generating the quests, both the Markov chain and the neural language model based generation systems use the object graph to determine object placement but only the Markov chain based model uses the action graph. This is because the instructions generated by the Markov chain model is in the form of a sequence of ingredients which then requires the action graph to determine the actions and additional objects required to turn this list of ingredients into a playable quest. The action graph would thus take an ingredient such as a carrot and determine first that it needs to be cut and that a knife is required for this task. The neural language model on the other hand already generates the full action, including potentially required objects, that can be executed and so does not make use of this graph.

5 Experiments

Our experiments were designed to compare perceived creativity and coherence, specifically testing our models in addition to factors such as complexity. We tested five types of designs: Human Designed (HD), Random Assignment (RA), Markov Chains Simple (MCS), Markov Chains Complex (MCC), and Language Model (LM). HD is simply what it sounds like, a game that was created by a person. In this game, a human creates both the ingredients and the instructions for a recipe and is additionally responsible for quest assembly, i.e. grounding the generated content in a given game world. We do not consider experience in designing text-adventure games when picking a human to create this game as this task can be performed even by novices given the easily understandable “home” theme of the game world. The game is manually crafted in terms of decided what ingredients to put where and what the final recipe would come together to be. RA is on the opposite end of the spectrum where, as the name suggests, everything is placed in a random location, and the recipe could be totally random with ingredients and instructions that might not normally be seen. MCS and MCC use our Markov chains approach to generate quest content. The difference between MCS and MCC are that the former has four ingredients involved in its recipe while the latter has eight. This was to vary the complexity to see how that affected perceived creativity. LM refers to the games generated using the recipes generated by the language model. We additionally had one-room and five-room variants for each of the models to test how the structure and length of the game would affect the players.

Evaluating the creativity of the output of any computational generation process is a difficult task which requires concrete definitions of the metrics being used. We thus setup the experiment by having our game designs deployed on Amazon Mechanical Turk for people to play and provide feedback. Specifically, they would play one randomly selected game from the 1 room layout and then fill out a survey for that game, and then play one randomly selected game from the 5 room layout and fill out an identical survey. Subjects were provided with a simple practice game that they could play beforehand to familiarize themselves with TextWorld and its interface. We had 75 total participants for the entire study and had an average of 15 people play each game. The only restrictions that we had for participants was that they had to be fluent in English—this was determined by means of prebuilt restrictions on Amazon Mechanical Turk and game completion verification.

The users were asked questions pertaining to two metrics: coherence and creativity. We looked at creativity as a metric in the survey using the components of creativity as defined by Boden: novelty, surprise, and value. The survey detailed

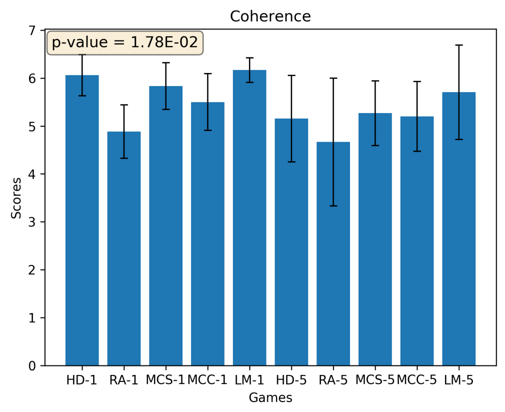


Figure 7: Coherence scores for each game. Error bars indicate one standard deviation.

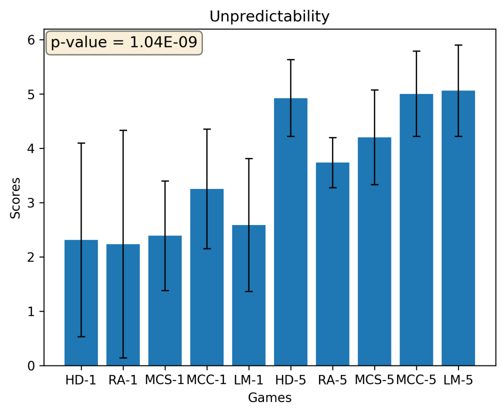


Figure 8: Unpredictability (surprise) scores for each game. Error bars indicate one standard deviation.

questions that measured our defined metrics, using Likert Scale values along a scale of 1-7. For example, it posed questions such as ”How original was the quest you played? 1: not at all novel, 7: exceptionally novel” when measuring novelty. The other factors were also measured using similarly phrased questions. A one-way ANOVA test was then conducted followed by Tukey HSD post-hoc analysis to determine significance. The results of the raw scores for each group as well as the significant results between pairs of different models are presented below.

6 Results and Discussion

We present results for four metrics: coherence, unpredictability (or surprise), novelty (or originality), and value (or accomplishment) for each of the games. Additionally, we also show the p-value result of a one way ANOVA test for the distributions in each of the categories to determine statistical significance. This test tells us if the differences in the means across the different games are significant for each of the categories separately. The Tukey HSD post-hoc analysis further tells us which specific pairs of results are significant. We hypothesized that semantic grounding using the knowledge graph would enable our models to maintain coherence on par with the human designed games. Further, given the stochastic nature of our generative models, we further predicted that our models would also rate as being comparable in terms of creativity to the human designed games—with all models relatively outperforming the randomly generated games. We see below that these predictions hold.

We find that the results for each individual category are significant—$p < 0.05$ in all the cases. Additionally, all the specific pairwise comparisons we make are significant with $p < 0.1$—a full set of these pairwise results can be found in Appendix A. The rest of this section will discuss each of these metrics in more detail.

Fig. 7 displays trends in the players' perception of coherence for each of the games. We first see that the one-room games were consistently rated to be more coherent than the five-room games, indicating that overall quest coherence—and thus the coherence of our generative system—degrades the longer and more complex the quest. Across the games, we see that the RA models were the considered to be the least coherent. The MCS model slightly outperforms the MCC model, showing that the Markov chain based models are more coherent the less complex the output. The LM achieves a higher score than both of the Markov chain models and maintains coherence more easily than either. Most importantly, all of these methods are comparable in coherence to the human-authored games, i.e. our semantically grounded knowledge graph ensures that coherence is not lost when generating content.

Similarly, Fig. 8 describes how surprising the game was to the players. The difference between the one-room and five-room games here is much more pronounced. The players find the five-room, the longer and more complex game, much more surprising than their one-room counterparts, showing that complexity is an important factor in determining surprise. Another indication of this is that the MCC model is rated as more surpris-

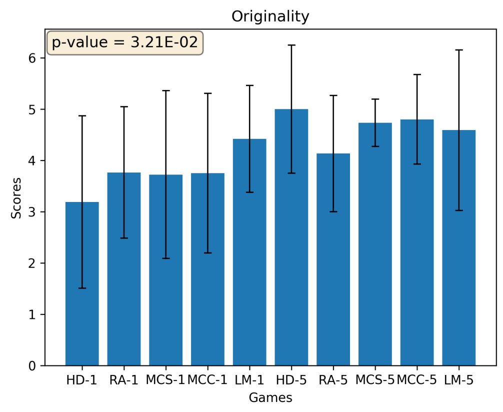


Figure 9: Originality (novelty) scores for each game. Error bars indicate one standard deviation.

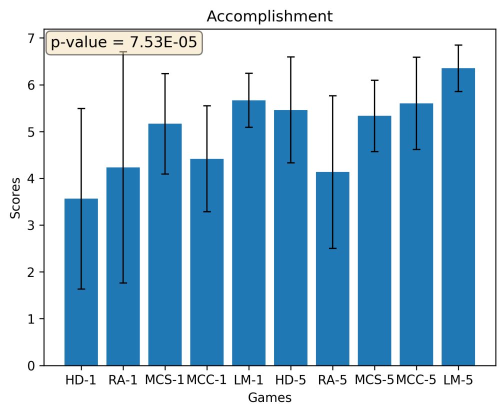


Figure 10: Accomplishment (value) scores for each game. Error bars indicate one standard deviation.

ing than the less complex MCS model. The LM achieves comparable performance to MCC and once again they all perform as well as the HD games.

Originality (Fig. 9), which we use as a proxy to measure novelty, exhibits similar trends as surprise. The more longer, more complex games are deemed more original. Despite being random, the RA games are seen to be less original than the the rest of the games perhaps indicating that there is a link between perceptions of coherence and originality. The gaps in performance here are much less pronounced, however, and the Markov chain models slightly edge out the LM — with all three being comparable to the HD games.

To measure value, or utility, in a text-adventure game, we asked the players if they felt a sense of accomplishment after finishing the game (Fig. 10). We see players reported a higher sense of accomplishment after finishing more complex games in general with the exception of the RA games, both of which performed poorly—likely due to them being relatively incoherent. We also note that the LM showed the highest values here, surpassing the HD games. We hypothesize that this might be due to the player having to perform a wider range of actions, some relatively unintuitive, that are not constrained by our action graph.

7 Conclusions

We have demonstrated a framework to automatically generate cooking quests in a "home" themed text-adventure game, although our framework can be generalized to other themes as well. Quest generation in a given game world is a subset of the overall problem of generating entire text-adventure games. Content generated by both the Markov chains and the neural language models can be grounded into a given game world using domain knowledge encoded in the form of a knowledge graph. The models each excel on different metrics: the Markov chains model produces quests that are more surprising and novel while the neural language model offers greater value and coherence. We also note, however, that the neural language model requires less domain knowledge than the Markov chains and is thus potentially more generalizable to other themes and types of quests.

Our human subject study shows us that there is an inverse relationship between creativity and coherence but only when a certain threshold of coherence is passed. In other words, the less coherent a game the more creative it is, but incoherent games—such as those generated by the RA model—are perceived to be less creative. Furthermore, our automatically generated games consistently perform at least as well as human designed games in this setting, both in terms of coherence and creativity—implying that the generative process can be automated without a loss in perceived game quality.

8 Acknowledgements

The authors would like to thank Dr. Ashok Goel and Fred Bane from the Cognitive Science class at Georgia Tech for their advice and encouragement of the ideas in this work.

References

Prithviraj Ammanabrolu and Mark O. Riedl. 2019a. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*.

Prithviraj Ammanabrolu and Mark O. Riedl. 2019b. Transfer in deep reinforcement learning using knowledge graphs. *CoRR*, abs/1908.06556.

Gabor Angeli, Johnson Premkumar, Melvin Jose, and Christopher D. Manning. 2015. Leveraging Linguistic Structure For Open Domain Information Extraction. 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)*.

Margaret A Boden. 2007. Creativity in a nutshell. *Think*, 5(15):83–96.

Antoine Bordes, Nicolas Usunier, Ronan Collobert, and Jason Weston. 2010. Towards understanding situated natural language. In *Proceedings of the 2010 International Conference on Artificial Intelligence and Statistics*.

Jerome Bruner. 1991. The narrative construction of reality. *Critical Inquiry*, 18(1):1–21.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2018. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*.

Allen I Fleishman. 1978. A method for simulating non-normal distributions. *Psychometrika*, 43(4):521–532.

Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. 2017. What can you do with a rock? affordance extraction via word embeddings. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1039–1045.

Spyridon Giannatos, Mark J Nelson, Yun-Gyung Cheong, and Georgios N Yannakakis. 2011. Suggesting New Plot Elements for an Interactive Story. In *In Workshop on Intellignet Narrative Technologies (INT'11)*.

Matthew Guzdial, Brent Harrison, Boyang Li, and Mark Riedl. 2015. Crowdsourcing open interactive narrative. In *FDG*.

Matan Haroush, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. 2018. Learning How Not to Act in Text-Based Games. In *Workshop Track at ICLR 2018*, pages 1–4.

K. Hartsook, A. Zook, S. Das, and M. O. Riedl. 2011. Toward supporting stories with procedurally generated game worlds. In *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*, pages 297–304.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2019a. Interactive fiction games: A colossal adventure. *CoRR*, abs/1909.05398.

Matthew Hausknecht, Ricky Loynd, Greg Yang, Adith Swaminathan, and Jason D. Williams. 2019b. Nail: A general interactive fiction agent. *CoRR*, abs/1902.04259.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep Reinforcement Learning with a Natural Language Action Space. In *Association for Computational Linguistics (ACL)*.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *CoRR*, abs/1904.09751.

Judith R Johnston. 2008. Narratives Twenty-Five Years Later. In *Top Lang Disorders*, volume 28, pages 93–98.

Boyang Li, Stephen Lee-Urban, Darren Scott Appling, and Mark O. Riedl. 2012. Crowdsourcing Narrative Intelligence . In *Advances in Cognitive Systems*, volume 1, pages 1–18.

Brian Magerko, Justin Permar, Mikhail Jacob, Margeaux Comerford, and Justin Smith. 2014. An Overview of Computational Co-creative Pretend Play with a Human. In *Proceedings of IVA 2014*.

Lara J. Martin, Prithviraj Ammanabrolu, Xinyu Wang, Shruti Singh, Brent Harrison, Murtaza Dhuliawala, Pradyumna Tambwekar, Animesh Mehta, Richa Arora, Nathan Dass, Chris Purdy, and Mark O. Riedl. 2017. Improvisational Storytelling Agents. In *Workshop on Machine Learning for Creativity and Design (NeurIPS 2017)*, Long Beach, CA.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language Understanding for Text-based Games Using Deep Reinforcement Learning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Justin Permar and Brian Magerko. 2013. A Conceptual Blending Approach to the Generation of Cognitive Scripts for Interactive Narrative. In *9th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE13)*.

Christopher Purdy, Xinyu Wang, Larry He, and Mark Riedl. 2018. Predicting Generated Story Quality with Quantitative Measures. In *In Proceedings of 14th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE18)*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Ronen Tamari, Hiroyuki Shindo, Dafna Shahaf, and Yuji Matsumoto. 2019. Playing by the book: An interactive game approach for action graph extraction from text. In *Proceedings of the Workshop on Extracting Structured Knowledge from Scientific Publications*, pages 62–71, Minneapolis, Minnesota. Association for Computational Linguistics.

Ruo Yu Tao, Marc-Alexandre Côté, Xingdi Yuan, and Layla El Asri. 2018. Towards solving text-based games by producing adaptive action spaces. In *Proceedings of the 2018 NeurIPS Workshop on Wordplay: Reinforcement and Language Learning in Text-based Games*.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. *CoRR*, abs/1903.03094.

Tony Veale, Diarmuid O'donoghue, and Mark T Keane. 2000. Computation and Blending. *Cognitive Linguistics*, 11(3/4):253282.

Zhilin Yang, Saizheng Zhang, Jack Urbanek, Will Feng, Alexander H. Miller, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Mastering the dungeon: Grounded language learning by mechanical turker descent. In *In proceedings of the The Sixth International Conference on Learning Representations (ICLR'18)*.

Alexander Zook, Brian Magerko, and Mark Riedl. 2011. Formally modeling pretend object play. In *Proceedings of the 8th ACM Conference on Creativity and Cognition*, pages 147–156, New York, NY, USA. ACM.

A Results

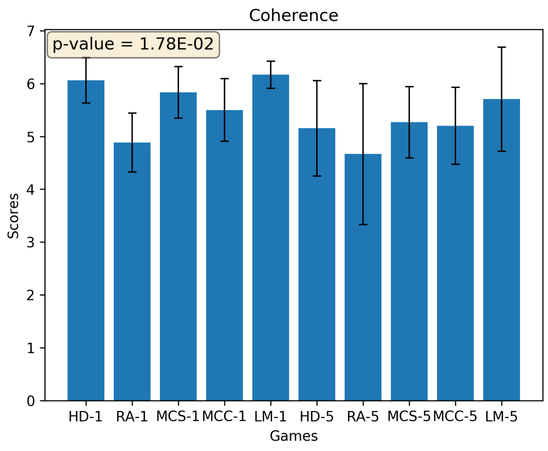


Figure 11: Coherence scores for each game. Error bars indicate one standard deviation.

Group 1	Group 2	Meandiff
HD-1	RA-5	-1.3958
LM-1	RA-5	-1.5
MCC-5	RA-5	-0.5333
MCS-1	RA-5	-1.1667
RA-1	RA-5	-1.2157
LM-5	MCS-1	0.1275
LM-5	MCS-5	-0.4392
LM-5	RA-5	-1.0392
LM-1	RA-1	-0.2843

Table 1: Coherence results for the post-hoc Tukey HSD test with $p = 0.1$, only significant results are shown.

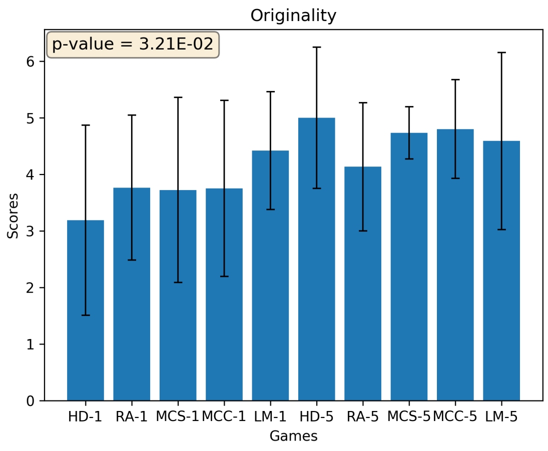


Figure 12: Originality (novelty) scores for each game. Error bars indicate one standard deviation.

Group 1	Group 2	Meandiff
HD-1	HD-5	1.8125
HD-1	LM-1	1.2292
HD-1	LM-5	1.4007
HD-1	MCC-5	1.6125
HD-1	MCS-1	0.5347
HD-1	MCS-5	1.5458
HD-1	RA-1	1.5772
HD-1	RA-5	0.9458
MCC-1	MCC-5	1.05
MCC-1	RA-1	1.0147

Table 2: Originality results for the post-hoc Tukey HSD test with $p = 0.1$, only significant results are shown.

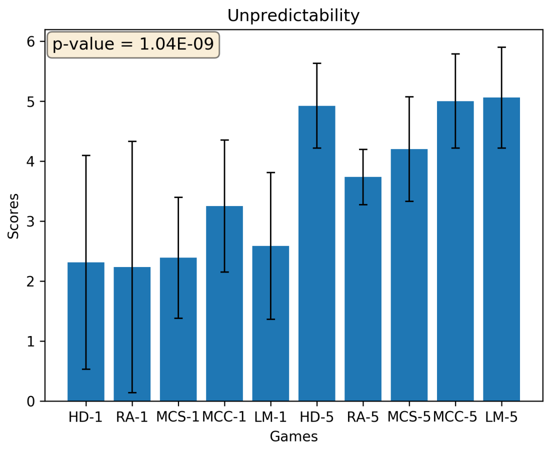


Figure 13: Unpredictability (surprise) scores for each game. Error bars indicate one standard deviation.

Group 1	Group 2	Meandiff
HD-1	HD-5	2.6106
HD-1	LM-5	2.7463
HD-1	MCC-5	2.6875
HD-1	MCS-5	1.8875
HD-1	MCC-1	0.9375
HD-1	RA-1	0.9228
HD-1	RA-5	1.4208
HD-5	LM-1	-2.3397
HD-5	MCC-1	-1.6731
HD-5	MCS-1	-2.5342
LM-1	LM-5	2.4755
LM-1	MCC-1	0.6667
LM-1	MCC-5	2.4167
LM-1	MCS-5	1.6167
LM-5	MCC-1	-1.8088
LM-5	MCS-1	-2.6699
LM-5	RA-1	-1.8235
LM-5	RA-5	-1.3255
MCC-1	MCC-5	1.75
MCC-1	MCS-1	-0.8611
MCC-5	RA-1	-1.7647
MCC-5	RA-5	-1.2667
MCS-1	MCS-5	1.8111
MCS-1	RA-1	0.8464
MCS-1	RA-5	1.3444

Table 3: Surprise results for the post-hoc Tukey HSD test with $p = 0.1$, only significant results are shown.

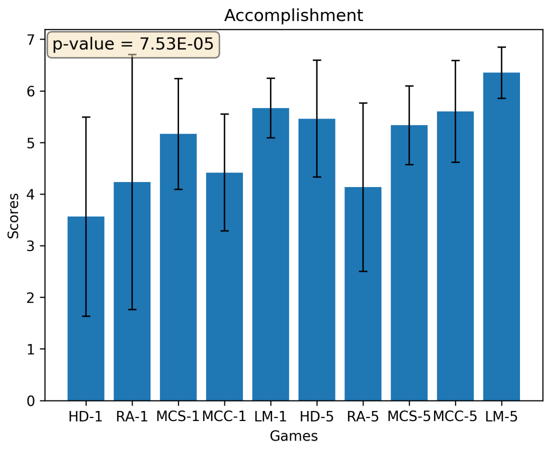


Figure 14: Accomplishment (value) scores for each game. Error bars indicate one standard deviation.

Group 1	Group 2	Meandiff
HD-1	HD-5	1.899
HD-1	LM-1	2.1042
HD-1	LM-5	2.7904
HD-1	MCC-5	2.0375
HD-1	MCS-1	1.6042
HD-1	MCS-5	1.7708
HD-1	RA-1	1.6728
HD-5	LM-5	0.8914
HD-5	RA-5	-1.3282
LM-1	MCC-1	-1.25
LM-1	MCS-1	-0.5
LM-1	RA-5	-1.5333
LM-5	MCC-1	-1.9363
LM-5	MCS-1	-1.1863
LM-5	MCS-5	-1.0196
LM-5	RA-1	-1.1176
LM-5	RA-5	-2.2196
MCC-1	MCC-5	1.1833
MCC-5	RA-5	-1.4667
MCS-5	RA-5	-1.2

Table 4: Value results for the post-hoc Tukey HSD test with $p = 0.1$, only significant results are shown.

Text Embellishment using Attention Based Encoder-Decoder Model

Subhajit Naskar
University of Massachusetts
Amherst, MA, US
snaskar@cs.umass.edu

Soumya Saha
University of Massachusetts
Amherst, MA, US
soumyasaha@cs.umass.edu

Sreeparna Mukherjee
University of Massachusetts
Amherst, MA, US
sreeparnamuk@cs.umass.edu

Abstract

Text embellishment is a natural language generation problem that aims to enhance the lexical and syntactic complexity of a text. i.e., for a given sentence, the goal is to generate a sentence that is lexically and syntactically complex while retaining the same semantic information and meaning. In contrast to text simplification (Wang et al., 2016), text embellishment is considered to be a more complex problem as it requires linguistic expertise, and therefore are difficult to be shared across different platforms and domain. In this paper, we have explored this problem through the light of neural machine translation and text simplification. Instead of using a standard sequential encoder-decoder network, we propose to improve text embellishment with the Transformer model. The proposed model yields superior performance in terms of lexical and syntactic embellishment and demonstrates broad applicability and effectiveness. We also introduce a language and domain agnostic evaluation set up specifically for the task of embellishment that can be used to test different embellishment algorithms.

1 Introduction

In recent years, deep neural networks have achieved some promising results in natural language tasks such as speech recognition, text generation, and machine translation. (Kim et al., 2015); (Zaremba et al., 2014); (Mikolov et al., 2010). (Rajeswar et al., 2017). Many of these models follow a *teacher forcing* technique, where the model is trained to predict the next word in the sequence given the previous words. This is usually done using maximum-likelihood training of these models. These models are then evaluated based on sequence level metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), etc.

Narrative generation or story generation is a common natural language generation task. Narrative generation pipeline consists of two steps (Yao et al., 2018): First step is to generate a simplified story by a human or a machine learning model. The second step mainly consists of discourse generation, i.e., producing narratives that sound meaningful and appealing to the readers. Our model, which performs text embellishment, can be used in the second step mentioned above. We use the word ”Embellishment” as a method to produce engaging texts out of simplified texts.

Previous researches on similar tasks have mainly dealt with rule-based approaches for discourse, where the model designer has predefined these rules. These rule-based approaches require significant expertise of the language and, more specifically, the field of the text. Furthermore, there is no scope of model generalization, i.e., a model for one type of text may not be used for a different type. Therefore, a domain-independent model for text embellishment has significant importance in the field of natural language generation, particularly narrative generation. A domain-independent model allows the designer to build a light-weight system to generate simple text and use a domain-independent text embellishment model to make their model more diverse in terms of sentence complexity.

The embellishment characteristic of a sentence can be categorized into two types,

- Lexical embellishment: The intent behind lexical embellishment is to replace commonly occurring vocabulary words with their complex counterparts while keeping the overall meaning of the sentence undisturbed. Here complexity is judged based on a similar methodology as *Word Complexity Measure* used for phonological assessment.

- Syntactic embellishment: Syntactic embellishment targets to increase the complexity of a sentence as a whole. This includes both grammatical and structural complexity enhancement.

In this paper, we primarily focus on lexical embellishment, i.e., for a given sentence, we try to generate grammatically correct sentence that has more complex words without changing the meaning of the sentence while exploring the possibility of syntactical embellishment.

The text embellishment task is often interpreted as the inverse to text simplification (TS), which has significant work and literature. Recent works on text simplification systems are capable of simplifying text both lexically and syntactically independent of domains or any predefined rules. Thus, we were encouraged to explore the possibility of a domain-independent text embellishment. However, we do acknowledge that text embellishment is generally a significantly complicated task compared to text simplification. Text simplification often can be regarded as a careful information reduction process, whereas text embellishment requires language generation. So, defining text embellishment as merely the inverse task of text simplification can be misleading.

Instead, we would like to argue that text embellishment shares certain traits with machine translation. Since our goal is to embellish sentences by replacing simple words with more complex words, our goal can be interpreted as a translation task where the source language is **simple** English and the target language is **complex** English.

While exploring past machine learning and natural language literature, we found that domain-independent text embellishment is a relatively unexplored task and seems exciting and promising. Furthermore, there is no prior work on domain-independent text embellishment that was able to show promising results in terms of either lexical embellishment or syntactic embellishment. In this paper, we solely explore the possibility of neural encoder-decoder architecture in developing a domain-agnostic text embellishment system. We use Transformer based architecture to improve embellishment quality and compare the results with seq2seq based architecture used on the same task.

2 Related Work

Research in computational narrative traces back to the 1960s and 1970s, intending to instill narrative intelligence in machines. One of the most well-known generation systems is TALE-SPIN, which produces narratives by emphasizing problem-solving techniques (Meehan, 1977). While this work focused on the idea that events follow each other sequentially, the work of (Callaway and Lester, 2002) explicitly addressed the gap between computational narrative and Natural Language Generation. These works use rule-based language models that produce naturally sounding narratives. Even though these approaches make use of text embellishment, the main disadvantage is that these rules have to be devised before the system's architecture design, which limits the performance of the model.

Another close area of research in this domain is incorporation of linguistic style. Here, style can refer to features of lexis, grammar, and semantics, which is individual to a particular author or a specific situation. While research in this area started with rule-based methods, but from the early 2000s, the shift has been towards a data-driven approach. (Paiva and Evans, 2005) developed an algorithm that identifies a series of local decisions that maximize the desired stylistic capacity. More recently, (Ficler and Goldberg, 2017) demonstrate controlling several stylistic variations in generated text through conditioned language models. On the contrary, we are trying to make the machine learn natural language representation from human data and enhance narratives that have been generated before in a domain-independent manner.

This approach can be compared with other existing works in the field of natural language processing such as statistical machine translation, text summarization, and, most importantly, text simplification. If we look at the summarization and simplification task, we will understand that the main goal is to extract the necessary information, maintain a natural language structure and remove linguistic embellishment that is not necessary for understanding the context. Thus, we see that our work is complementary in its objective to both of these tasks. Thus, the applications of text simplification include reducing the complexity of a natural language sentence by focusing on the discourse level aspects of syntactic simplification (Siddharthan, 2002), whereas on the other hand

(Coster and Kauchak, 2011) aims to reduce the reading complexity of a sentence by incorporating more accessible vocabulary and sentence structure. Thus, here in our task is more similar to the later as we aim to increase the complexity of a simple sentence by adorning its vocabulary with more complex words. Another prominent work is this field by (Shardlow, 2014), where they distinguish between syntactic and lexical simplification. Syntactic simplification aims to reduce the complexity of sentence structure, whereas lexical simplification aims to replace difficult vocabulary with simpler words. Previously, these two types of simplification tasks have been addressed separately. ((Siddharthan, 2006), (Biran et al., 2011), (Paetzold and Specia, 2017)). More recently, we have seen that both these tasks have been addressed simultaneously ((Wang et al., 2016), (Zhang and Lapata, 2017)). These tasks address this problem as an extension of machine translation and borrow ideas from automatic natural language generation (Wen et al., 2015). So, the problem of simplification comes down to monolingual machine translation, where the goal is to translate from a **complex** English to a **simple** English. Following the recent success of neural machine translation, we see an increased use of LSTM based encoder-decoder architecture in these tasks. ((Bahdanau et al., 2014), (Cho et al., 2014)). Work by (Hochreiter and Schmidhuber, 1997) on Long Short-term Memory architecture has long been used to solve sequence to sequence tasks where both the input and output sequence can be of varied length.

Although we have seen significant progress in the domain of text simplification, little work has been done in text embellishment. What makes text embellishment promising right now are the vast corpora on which text simplification tasks have been trained for more than two decades. However, text embellishment is a much more difficult task than text simplification as adding of information might lead to an introduction of semantic contradictions.

Nevertheless, quite recently, a promising work in this field has been done by (Berov and Standvoss, 2018), motivated from researches in text simplification and machine translation. The authors have proposed a network design similar to what (Wang et al., 2016) have used in their text simplification work, which uses a Long Short-Term Memory (LSTM) Encoder-Decoder model for sentence-level text simplification as it makes minimal assumptions about word sequence. We replicate their model and use it as our baseline model for evaluation and experimentation.

In this paper, we focus on neural encoder-decoder architecture in developing a domain-agnostic text embellishment system. To our knowledge, there are no existing linguistically motivated, non-neural architecture for text embellishment. However, one can see how a word or phrase substitution method can be employed using a pre-defined mapping between simple word/phrase to complex word/phrase. This can be done using Wordnet (Miller, 1995) as shown by (Tambe et al., 2019) in context of text simplification. However, such substitution may lead to incorrect substitution in the context of a specific domain. As, a substitution that might be valid for a literary text(novel, short story) may be incorrect and misinforming for a different domain such as a scientific journal. That will necessitate domain-specific rule design. As discussed in (Tambe et al., 2019), Such methodology requires additional steps such as word sense disambiguation, lexical simplification, which are out of the scope of this paper. Furthermore, such substitution will limit the possibility of syntactic embellishments, such as structural and grammatical complexity enhancement. Therefore, we avoid discussing the substitution based embellishment strategy in this paper.

3 Methodology

We achieve the goal of embellishing a sentence by modeling the distribution of the embellished sentence given the simple sentence. i.e. $P(Y|X)$ where X denotes the simple sentence and the words of the simple sentence is denoted as $x_1, x_2, ..., x_n$. Similarly, Y denotes the embellished sentence, and the words of the embellished or target sentence are $y_1, y_2, ..., y_m$. We model our task similar to a machine translation task and employ an encoder-decoder architecture. The encoder consumes the input text and computes a representation of context vector c. The decoder generates one target word given the context vector c and all the previous predicted words $\bar{y_1}, ..., \bar{y_{t-1}}$.

$$p(y) = \prod_{t=1}^{T} p(y_t|\bar{y_1}, ..., \bar{y_{t-1}}, c) \qquad (1)$$

For both of our models, we used named entity masking and byte pair encoding in input sentences and beam search decoding while generating embellished sentences.

3.0.1 LSTM Architecture

From the work of (Wang et al., 2016), we can see that the LSTM Encoder-Decoder model can learn operational rules such as reversing, sorting, and replacing from sequence pairs. This shows such Encoder-Decoder model may potentially apply rules like modifying sentence structure, substituting words, and removing words for text simplification as well as text embellishment.

We chose our model as 3 LSTM layers having 300 hidden units for each encoder and decoder. All weights were uniformly initialized as [-0.1, 0.1]. We have used Harvards OpenNMT PyTorch framework (Klein et al., 2017) to set up the above network and used the model for our task. We have used a system having 8 core CPU with 2 NVIDIA P100 GPU to train our network.

3.1 Transformer architecture

The LSTM based encoder-decoder model operates sequentially using recurrence. Compared to sequence to sequence models, the Transformer processes all words or symbols in the sequence in parallel while making use of a self-attention mechanism to incorporate context from words farther away from it. By processing all words in parallel and letting each word attend to other words in the sentence over multiple processing steps, the Transformer is computationally much more efficient and gives superior performance in many natural language processing tasks. However, in our case, we will focus on the sequential nature of LSTMs and limitations. i.e., it is prone to be inferior in handling long term dependencies (even with attention). However, in our case, we require an architecture that is capturing semantic information and long term dependencies effectively. This motivated us to use the transformer model as it comes out as a promising architecture to address this problem.

The architecture we used in our implementation consists of 6 identical layers for each encoder and a decoder network with all the sublayers having 512 units and 8 parallel attention layers or heads. For our best performing model, we used Byte-pair-encoding on the input text. The model was trained on 2 NVIDIA GTX 1080 Ti GPUs.

4 Datasets

To train the model, we are using the WikiLarge dataset, constructed by (Zhang and Lapata, 2017). The dataset consists of 256252 aligned sentences for training, 854 aligned sentences for validation, and 358 sentences for test. We have chosen this particular dataset, primarily because this is the largest sentence-aligned dataset which is widely used for text simplification task. ((Xu et al., 2016) , (Vu et al., 2018) (Zhao et al., 2018))

Thus, since this task is complementary to text simplification, we have interchanged the source and target datasets, and now the goal of the model will be to produce ”complex” sentences from the ”simple” input sentences.

5 Results

The LSTM encoder-decoder model was able to achieve some basic lexical replacements (*found* $\rightarrow$ *discovered*, *stayed* $\rightarrow$ *remained*) and grammatical corrections such as character case-correction, punctuation correct. For example, *It was* ***found*** *by PERSON@1 ...* $\rightarrow$ *It was* ***discovered*** *by PERSON@1 ...*, ***the*** *former district PERSON@1 ...* $\rightarrow$ ***The*** *former district PERSON@1....*

It is to be noted, such instances of lexical embellishment were relatively limited, and in most cases, the output is identical to the input sentence. and there were no such instances of syntactic embellishment.

However, in the case of the Transformer model, there was a significantly high number of lexical embellishment. Also, along with one-one lexical replacement(replacing a single word with a more complex synonym), Transformer was able to replace POS phrases with more complex word(*very very* $\rightarrow$ *extremely*). Which was more impressive and noteworthy was, in some instances, we observed syntactic embellishment as well.

5.1 Lexical embellishment

Entrance to LOCATION@1 is ***very very*** *difficult* $\rightarrow$ *Entrance to LOCATION@1 is* ***extremely*** *difficult.*

Their culture ***is similar to*** *the culture ... The culture of LOCATION@1* ***is closely associated with*** *the culture ...*

5.2 Syntactic embellishment

It is a starting point *for people wanting ...* $\rightarrow$ ***It also serves as a starting point*** *for people wanting*

...,

... *appears as* ***a stretched*** *object . A stretched object]**was the major axis .It pointing towards Uranus*** $\rightarrow$ *appears as* ***an elongated*** *object,with the* ***major axis pointing towards Uranus***

From this example, we can see that our proposed model was able to achieve a more complex type of lexical embellishment and some impressive syntactic embellishment.

6 Evaluation

Primarily, we evaluate our models using two different evaluation setup: BLEU, readability scores. However, these standard metrics have certain limitations and may not always be sufficient for evaluating the embellishment capability of a model. Therefore, we design a human evaluation setup that is suitable for our task.

6.1 BLEU

To measure the proximity of generated output sentence's context to the original sentence, we have used BLEU score, which is an automated method to evaluate machine translation tasks. The main purpose of BLEU metric is to evaluate the *closeness* of a machine-generated translation with reference to its human translation. Now, in the context of our task, we use this metric to evaluate if the context of the sentence has been changed or not. Thus, a high BLEU score would indicate that the context is close or similar to the original sentence, whereas a low BLEU score would indicate that context has changed. One point to be noted is that this evaluation metric does not measure the level of embellishment in the hypothesis sentence compared to the reference sentence. In the following table, we present the BLEU scores of the best models we have trained.

Measure	LSTM	Transformer
BLEU	91.04	66.10

Table 1: BLEU with source sentence

From the high BLEU score of LSTM network, it can be inferred that LSTM network mainly learned to reproduce the input correctly without modifying any words in the source sentence. While the Transformer model produces sentences with a lower BLEU score, it does not provide us any significant information regarding its capability for embellishment. Furthermore, BLEU is often considered unsuitable and controversial for the language generation task, as argued by (Reiter, 2018). Thus, we have used Readability Measurements for that purpose.

6.2 Readability measures

To measure the complexity of the generated sentences, we are using three measures based on the readability of text. We have evaluated the generated sentences of both our models using these three measures, *Flesch Reading Ease score (FRES)*, *Flesch-Kincaid Grade Level (FKGL)* and *SMOG Index*. More details and corresponding equations used for each of the Readability Measures have been described in the Appendix section. In the following diagram, we have shown the readability statistics of the output of the LSTM and Transformer. We evaluate our test dataset based on the three different readability metrics and record the percentage of sentences where the readability scores increased, decreased, or stayed the same after embellishment.

From, figure 1, we can see that for all readability metrics, the percentage of data where the input and output sentence has the same readability score is significantly high in the case of LSTM. Which, further confirms the result reported by (Berov and Standvoss, 2018), that LSTM model is prone to copying the input to output without achieving any embellishment. However, if we compare the ratio of data with increased readability level, we see, Transformer model shows better embellishment performance. However, the percentage of data with readability is also high for Transformer. To inspect that phenomenon, we employ human evaluation and design our task-specific evaluation setup that will shed more light on this issue.

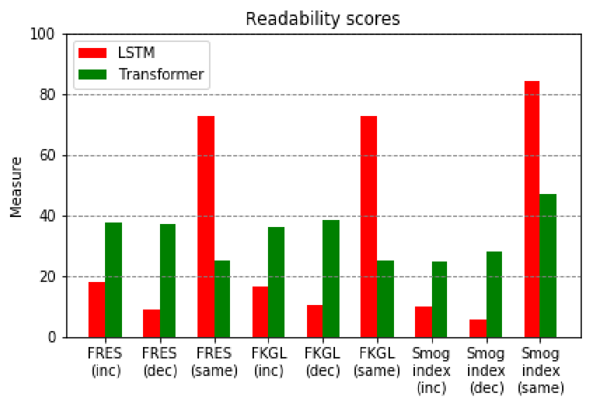


Figure 1: Readability Scores

6.3 Human evaluation

The shortcoming and limitations of the aforementioned evaluation metrics motivate a human evaluation process. For human evaluation, we designed metrics to score generated results. We employed 12 human evaluators who are proficient in English and are capable of evaluating the complexity of a text. Evaluators were randomly assigned to a group of 4, and 3 groups were formed to score model-generated results based on the scoring metrics. Then we used this scores to devise 5 model performance metrics, namely contextual capacity, generative capacity, consistency measure that measure a different aspect of how contextually coherent and meaningful the results are, and embellishment capacity and conditional embellishment capacity that measures the model's ability to embellish a given simple sentence. First, we will define the scoring metrics used by the evaluators to score the generated results. Then we will discuss the performance metrics.

6.3.1 Score metrics

The underlying goal of this task to design a system that can generate a meaningful, contextually identical, and embellished sentence for a given input sentence. Therefore, we designed scoring metrics to capture exactly that. We asked our evaluator to score each generated sentence on a categorical scale of 0,1,2 for the following three metrics.

- **Grammatical Coherence score**: If embellished sentences were grammatically correct and are a meaningful sentence.
- **Context Coherence score**: If the embellished sentence is within the same context of the simple source sentence.
- **Embellishment score**: If the generated sentence is overall more complex than the source sentence. If the model achieves generate a structurally complex sentence, that will be considered a successful embellishment. If the model manages to replace words, We asked participants to evaluate each word replacement based on whether the embellished words were more complex and if such replacement is leading to the embellished sentence becoming more complex.

6.3.2 Aggregate Performance metrics

The aforementioned scoring metrics are used to calculate the performance metrics defined below.

- **Contextual Capacity**: Model's capacity to generate contextually correct sentences.
- **Generative Capacity**: Model's capacity to generate contextually and grammatically correct sentences.
- **Embellishment Capacity**: Model's capacity to generate embellished sentences.
- **Conditional Embellishment Capacity**: Model's capacity to achieve embellishment given that the model generates contextually and grammatically correct sentences.
- **Consistency Measure**: Model's consistency of generating embellished sentence or the same sentence was given that the model always generates contextually and grammatically correct sentences.

For the sake of brevity, the categorical definition of scoring metrics and calculation procedure of performance metrics are documented in the Appendix.

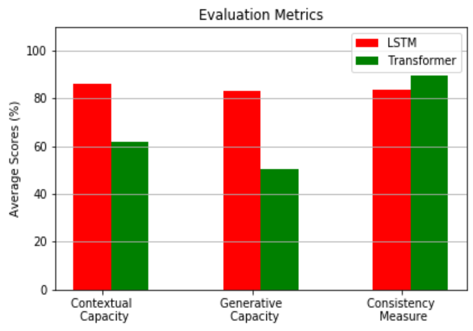


Figure 2: Evaluation metrics

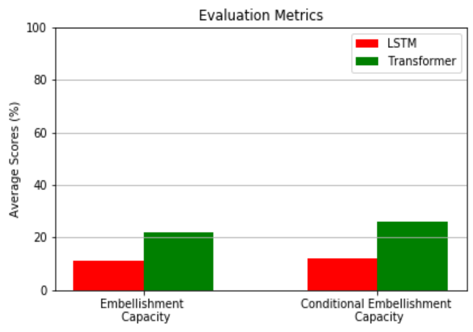


Figure 3: Evaluation metrics

From Figure 2, we can see that the LSTM model generates more contextually and grammatically consistent sentences, i.e., there are significantly fewer cases of random embellished output. The reason being, LSTM model, fails gracefully, i.e., when it fails to generate a lexically or syntactically complex sentence, it simply copies the input words to the output sentence. This same phenomenon was also recorded by (Berov and Standvoss, 2018) as well. Moreover, that is where we were able to achieve significant improvement. In figure 3, if we compare the embellishment capacity, the Transformer is significantly better (nearly double embellishment capacity) compared to LSTM encoder-decoder model. If we condition our evaluation on grammatically and contextually consistent outputs only, the transformer model outperforms LSTM encoder-decoder significantly. This may indicate that, when Transformer model can generate a contextually and grammatical consistent sentence, it has a significantly better power to achieve text embellishment.

7 Conclusions

Text embellishment is quite an unexplored track in the research field of Natural Language Generation because it would require a massive amount of data, training hours as well as various idiosyncratic, hand-coded rules to get performance which is close to human efficiency.

In this paper, the results from the LSTM encoder-decoder or the transformer network cannot be used for production purposes yet. Maybe, it is because the network is not able to learn all the nuances of human languages in a specific domain with the help of a dataset that is flawed with few grammatical and typographical errors. The availability of sentence aligned massive datasets that are more apt for this specific task is also rare.

In this paper, we show that our LSTM network achieved a BLEU score of 91.04, which is closer (92.13) to what (Berov and Standvoss, 2018) achieved with similar LSTM architecture. However, we showed why such measurements can be misleading and which motivated us to design a task-specific human evaluation setup. Based on the evaluation setup, we showed how transformer architecture is significantly better for the text embellishment task. Thus our initial assumption that in the case of generation tasks, especially text embellishment, a self-attention based architecture performs better than a seq2seq model with attention, holds. This is because these models are more capable of capturing the semantic information of a sentence.

8 Future work

In this paper, we aimed to work on the task of text embellishment for a single sentence. The same methodology and architecture can be extended to paragraphs, where we intend to generate a lexically and syntactically complicated paragraph given a simple paragraph. Such a task may require a hierarchical attention mechanism where we attend to single words as well as sentences to capture the semantics of a sentence and paragraph.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Leonid Berov and Kai Standvoss. 2018. Discourse embellishment using a deep encoder-decoder network. *arXiv preprint arXiv:1810.08076*.

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: A context-aware approach to lexical simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 496–501, Stroudsburg, PA, USA. Association for Computational Linguistics.

Charles B. Callaway and James C. Lester. 2002. Narrative prose generation. *Artif. Intell.*, 139(2):213–252.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

William Coster and David Kauchak. 2011. Simple english wikipedia: A new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 665–669, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. *CoRR*, abs/1707.02633.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

James R. Meehan. 1977. Tale-spin, an interactive program that writes stories. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'77, pages 91–98, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Gustavo Paetzold and Lucia Specia. 2017. Lexical simplification with neural ranking. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 34–40.

Daniel S. Paiva and Roger Evans. 2005. Empirically-based control of natural language generation. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 58–65.

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 '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sai Rajeswar, Sandeep Subramanian, Francis Dutil, Christopher Joseph Pal, and Aaron C. Courville. 2017. Adversarial generation of natural language. *CoRR*, abs/1705.10929.

Ehud Reiter. 2018. A structured review of the validity of BLEU. *Computational Linguistics*, 44(3):393–401.

Matthew Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4.

Advaith Siddharthan. 2002. An architecture for a text simplification system. In *Proceedings of the Language Engineering Conference (LEC'02)*, LEC '02, pages 64–, Washington, DC, USA. IEEE Computer Society.

Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.

Mrunmayee Tambe, Preeti Ballal, Vishal Dolase, Kajol Agrawal, and Yogesh Rajmane. 2019. *Lexical Text Simplification Using WordNet*, pages 114–122.

Tu Vu, Baotian Hu, Tsendsuren Munkhdalai, and Hong Yu. 2018. Sentence simplification with memory-augmented neural networks. *CoRR*, abs/1804.07445.

Tong Wang, Ping Chen, Kevin Michael Amaral, and Jipeng Qiang. 2016. An experimental study of LSTM encoder-decoder model for text simplification. *CoRR*, abs/1609.03663.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Peihao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *CoRR*, abs/1508.01745.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2018. Plan-and-write: Towards better automatic storytelling. *CoRR*, abs/1811.05701.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605. Association for Computational Linguistics.

Sanqiang Zhao, Rui Meng, Daqing He, Andi Saptono, and Bambang Parmanto. 2018. Integrating transformer and paraphrase rules for sentence simplification. *CoRR*, abs/1810.11193.

9 Appendix

9.1 Human evaluation method

The readability measurements are apt for evaluation of human-generated embellishments, but they often fail in case of machine-generated text embellishments. Thus to provide leniency considering the capabilities of various deep learning architectures in this particular task, we have decided to evaluate our model using a small survey. We believe human judgment will be the best in understanding the level of embellishments in sentences produced using our methods. These groups of judges were asked to score the generated sentences based on lexical embellishment and grammatical and context coherence.

- Grammatical Coherence: Each participant were asked to judge if the embellished sentences were grammatically correct and is a meaning sentence.

Score	Interpretation
0	Not grammatical and meaningful
1	Grammatically partially correct
2	Grammatically correct

Table 2: Rubric for Grammatical coherence score

- Context Coherence: Each participant were asked to judge if the embellished sentence is within the same context of the simple source sentence.

Score	Interpretation
0	Deviated from context
1	Partially correct context
2	Correct context

Table 3: Rubric for Context coherence score

- Embellishment: We asked each participant to evaluate each word replacement based on whether the embellished words were more complex or the embellished sentence was more complex. Each participants assigned a score between 0 to 2 for each sentence.

Score	Interpretation
0	Not meaningful sentence
1	Same sentence or hard to decide
2	Embellishment

Table 4: Rubric for Embellishment score

For ease of evaluation, we also asked the participants to judge the source or simple sentences based on their grammatical correctness, sentence structure to decide if the sentence is a correct English sentence and has any scope of textual embellishment.

Score	Interpretation
0	Grammatically incorrect
1	Confusing or conveys no meaning
2	Proper English sentence

Table 5: Rubric for source score

This source score will help us record the number of sentences in the test data that can be embellished. However, For human evaluation based measurements we will discard all sentences with source score of 0 and 1.

Based on these scores, we recorded the following:

- Context-wise correct: Number of embellished sentences with context coherence score(C) of 2, denoted as $f(C = 2)$.
- Grammatically and context-wise correct: Number of embellished sentences with context coherence score(C) 2 and grammatical coherence score(G) 2, denoted as $f(C = 2, G = 2)$.
- Grammatically, context-wise correct and good embellishment: Number of embellished sentences with context coherence score(C) 2 and grammatical coherence score(G) 2 and embellishment score 2, denoted as $f(C = 2, G = 2, E = 2)$
- Grammatically, context-wise correct and partial embellishment: Number of embellished sentences with context coherence score(C) 2 and grammatical coherence score(G) 2 and embellishment score 2, denoted as $f(C = 2, G = 2, E = 1)$

Based on this, we will derive the following performance measures where N is the total number of test sentences used:

- **Contextual Capacity**: Model's capacity to generate contextually correct sentences.

$$Contextual\ Capacity = \frac{f(C = 2)}{N}$$

- **Generative Capacity**: Model's capacity to generate contextually and grammatically correct sentences.

$$Generative\ Capacity = \frac{f(C = 2, G = 2)}{N}$$

- **Embellishment Capacity**: Model's capacity to generate embellished sentences.

$$Embellishment\ Capacity = \frac{f(E = 2)}{N}$$

- **Conditional Embellishment Capacity**: Model's capacity to achieve given that the model always generates contextually and grammatically correct sentence.

$$Conditional\ Embellishment\ Capacity = \frac{f(C=2,G=2,E=2)}{f(C=2,G=2)}$$

- **Consistency Measure**: Model's consistency of generating embellished sentence or same sentence given that the model always generates contextually and grammatically correct sentence.

$$Consistency\ Measure = \frac{f(C=2,G=2,E=2)}{f(C=2,G=2)} + \frac{f(C=2,G=2,E=1)}{f(C=2,G=2)}$$

9.2 Readability Measures

- **Flesch Reading Ease score (FRES)**

  The Flesch Reading Ease score, developed by Rudolf Flesch, is the most commonly used readability measure. The score on the test will tell us roughly what level of education someone will need to be able to read a piece of text easily. The Reading Ease formula generates a score between 1 and 100. The formula for the Flesch reading ease score (FRES) test is

$$\text{FRES} = 206.835 - 1.015 * \frac{\text{total words}}{\text{total sentences}} - 84.6 * \frac{\text{total syllables}}{\text{total words}}$$

- **Flesch-Kincaid Grade Level (FKGL)**

  The Flesch-Kincaid Grade Level index is one way to measure and report the readability of English text. Both Flesch reading ease and Flesch-Kincaid grade level use the same core metrics: word length and sentence length. But they correlate inversely. If one receives a high score on the reading ease test, one should receive a lower grade level score. The FKGL formula presents a score as a U.S. grade level, making it easier for teachers, parents, librarians, and others to judge the readability level of various books and texts. It can also mean the number of years of education generally required to understand this text, relevant when the formula results in a number greater than 10. The grade level is calculated with the following formula:

$$\text{FKGL} = 0.39 * \frac{\text{total words}}{\text{total sentences}} + 11.8 * \frac{\text{total syllables}}{\text{total words}} - 15.59$$

- **SMOG Index**

  The SMOG Index is also a measure of readability that estimates the years of education needed to understand a piece of writing. What is different from the above two evaluation measures is it considers the number of polysyllables (words of 3 or more syllables) whereas the above two measures consider average number of syllables per sentence. The formula for calculating SMOG index is:

$$\text{SMOG} = 1.043 * \sqrt{\text{\# of polysyllables}} * \sqrt{\frac{30}{\text{total sentences}}} + 3.1291$$

9.3 Additional results

9.3.1 LSTM encoder-decoder

*It was **found** by PERSON@1 in images from the Voyager NUMBER@1 $\rightarrow$ It was **discovered** by PERSON@1 in images from the Voyager NUMBER@1*

*This stamp **stayed** the standard letter stamp for the rest of PERSON@1 's reign, and many were printed $\rightarrow$ This stamp **remained** the standard letter stamp for the rest of PERSON@1 's reign , and many were printed .*

*In the year NUMBER@1 ,the population was NUMBER@2 . $\rightarrow$ The population was NUMBER@1 at the NUMBER@2 **census** .*

*the former district PERSON@1 , also resembles the upper half of the coat of arms . $\rightarrow$ **The** former district PERSON@1 , also resembles the upper half of the coat of arms .*

In December , NUMBER@1 , PERSON@1 was honored as part of the Righteous Among the Nations by the State of LOCATION@1 . $\rightarrow$ In December NUMBER@1 , PERSON@1 was honored as part of the Righteous Among the Nations by the State of LOCATION@1 .

9.3.2 Transformer

It is a starting point *for people wanting to explore LOCATION@1 , LOCATION@2 and LOCATION@3 .* $\rightarrow$ ***It also serves as a starting point*** *for people wanting to explore LOCATION@1 , LOCATION@2 and LOCATION@3 .*

Their culture ***is similar to*** *the culture of the coastal peoples of LOCATION@1 .* $\rightarrow$ *The culture of LOCATION@1* ***is closely associated with*** *the culture of the coastal people of LOCATION@1*

entrance to LOCATION@1 is ***very very*** *difficult .* $\rightarrow$ *entrance to LOCATION@1 is* ***extremely*** *difficult .*

ORGANIZATION@1 ***named*** *him " Sportsman of the Year " in NUMBER@1 .* $\rightarrow$ *ORGANIZATION@1* ***crowned*** *him " Sportsman of the Year " in NUMBER@1.*

Early September NUMBER@1 , dry air wrapping around the southern area of the cyclone caused most of the heat to ***leave*** *.* $\rightarrow$ *Early September NUMBER@1 , dry air wrapping around the southern area of the cyclone caused most of the heat to* ***evacuate*** *.*

At the Voyager NUMBER@1 pictures PERSON@1 appears as ***a stretched object*** *. A stretched object was the major axis . It* ***pointing towards Uranus*** *.* $\rightarrow$ *At the Voyager NUMBER@1 pictures PERSON@1 appears as an* ***elongated object, with the major axis pointing towards Uranus*** *.*

Some ***clauses*** *are rather lengthy and rich in content while others are shorter -LRB- possibly stubs -RRB- and of lesser quality .* $\rightarrow$ *Some* ***language content*** *are rather lengthy in content while others are shorter -LRB- possibly stubs -RRB- and of lesser quality .*

In NUMBER@1 PERSON@1 was inducted into the Rock and ORGANIZATION@1 . $\rightarrow$ *In NUMBER@1* ***,*** *PERSON@1 was inducted into the Rock and ORGANIZATION@1 .*

Emotional Neural Language Generation Grounded in Situational Contexts

Sashank Santhanam and Samira Shaikh
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, NC, USA
{ssantha1,samirashaikh}@uncc.edu

Abstract

Emotional language generation is one of the keys to human-like artificial intelligence. Humans use different type of emotions depending on the situation of the conversation. Emotions also play an important role in mediating the engagement level with conversational partners. However, current conversational agents do not effectively account for emotional content in the language generation process. To address this problem, we develop a language modeling approach that generates affective content when the dialogue is situated in a given context. We use the recently released Empathetic-Dialogues corpus to build our models. Through detailed experiments, we find that our approach outperforms the state-of-the-art method on the perplexity metric by about 5 points and achieves a higher BLEU metric score.

1 Introduction

Rapid advancement in the field of generative modeling through the use of neural networks has helped advance the creation of more intelligent conversational agents. Traditionally these conversational agents are built using *seq2seq* framework that is widely used in the field of machine translation (Vinyals and Le, 2015). However, prior research has shown that engaging with these agents produces dull and generic responses whilst also being inconsistent with the emotional tone of conversation (Vinyals and Le, 2015; Li et al., 2016c). These issues also affect engagement with the conversational agent, that leads to short conversations (Venkatesh et al., 2018). Apart from producing engaging responses, understanding the situation and producing the right emotional response to a that situation is another desirable trait (Rashkin et al., 2019).

Emotions are intrinsic to humans and help in creation of a more engaging conversation (Poria et al., 2019). Recent work has focused on approaches towards incorporating emotion in conversational agents (Asghar et al., 2018; Zhou et al., 2018; Huang et al., 2018; Ghosh et al., 2017), however these approaches are focused towards seq2seq task. We approach this problem of emotional generation as a form of transfer learning, using large pretrained language models. These language models, including BERT, GPT-2 and XL-Net, have helped achieve state of the art across several natural language understanding tasks (Devlin et al., 2019; Radford et al., 2019; Yang et al., 2019). However, their success in language modeling tasks have been inconsistent (Ziegler et al., 2019). In our approach, we use these pretrained language models as the base model and perform transfer learning to fine-tune and condition these models on a given emotion. This helps towards producing more emotionally relevant responses for a given situation. In contrast, the work done by Rashkin *et al.* (2019) also uses large pretrained models but their approach is from the perspective of seq2seq task.

Our work advances the field of conversational agents by applying the transfer learning approach towards generating emotionally relevant responses that is grounded on emotion and situational context. We find that our fine-tuning based approach outperforms the current state of the art approach on the automated metrics of the BLEU and perplexity. We also show that transfer learning approach helps produce well crafted responses on smaller dialogue corpus.

2 Approach

Consider the example show in Table 1 that shows a snippet of the conversation between a speaker and a listener that is grounded in a situation representing a type of emotion. Our goal is to produce responses to conversation that are emotionally appropriate to the situation and emotion portrayed. We approach this problem through a language modeling approach. We use large pre-trained language model as the base model for our response generation. This model is based on the transformer architecture and makes uses of the multi-headed self-attention mechanism to condition itself of the previously seen tokens to its left and produces a distribution over the target tokens. Our goal is to make the language model $p(y) = p(y_1, y_2, ..., y_t; \theta)$ learn on new data and estimate the conditional probability $p(y|x)$. Radford *et al.* (2019) demonstrated the effectiveness of language models to learn from a zero-shot approach in a multi-task setting. We take inspiration from this approach to condition our model on the task-specific variable $p(y_t|x, y_{<t})$, where x is the task-specific variable, in this case the emotion label. We prepend the conditional variable (emotion, situational context) to the dialogue similar to the approach from Wolf *et al* (2019). We ensure that that the sequences are separated by special tokens.

Emotion: Confident **Situation:** I just knew I was going to do well at work this morning.
Speaker: I just knew I was going to do well at work this morning. I was prepared **Listener:** That is the way to go! Keep it up!

Table 1: Example of conversations between a speaker and a listener

3 Experiments

3.1 Data

In our experiments we use the Empathetic Dialogues dataset made available by Rashkin *et al.* (2019). Empathetic dialogues is crowdsourced dataset that contains dialogue grounded in a emotional situation. The dataset comprises of 32 emotion labels including *surprised, excited, angry, proud, grateful*. The speaker initiates the conversation using the grounded emotional situation and the listener responds in an appropriate manner[1].Table 2 provides the basic statistics of the corpus.

	Train	Valid.	Test
Num. Conversations	19433	2770	2547
Utterances	84324	12078	10973
Avg Length Conversations	4.31	4.36	4.31

Table 2: Statistics of Empathetic Dialogue dataset used in our experiments

3.2 Implementation

In all our experiments, we use the GPT-2 pre-trained language model. We use the publicly available model containing 117M parameters with 12 layers; each layer has 12 heads. We implemented our models using PyTorch Transformers.[2] The input sentences are tokenized using byte-pair encoding(BPE) (Sennrich et al., 2016) (vocabulary size of 50263). While decoding, we use the nucleus sampling ($p = 0.9$) approach instead of beam-search to overcome the drawbacks of beam search (Holtzman et al., 2019; Ippolito et al., 2019). All our models are trained on a single TitanV GPU and takes around 2 hours to fine-tune the model. The fine-tuned models along with the configuration files and the code will be made available at: `https://github.com/sashank06/CCNLG-emotion`.

3.3 Metrics

Evaluating the quality of responses in open domain situations where the goal is not defined is an important area of research. Researchers have used methods such as BLEU , METEOR (Banerjee and Lavie, 2005), ROUGE (Lin, 2004) from machine translation and text summarization (Liu et al., 2016) tasks. BLEU and METEOR are based on word overlap between the proposed and ground truth responses; they do not adequately account for the diversity of responses that are possible for a given input utterance and show little to no correlation with human judgments (Liu et al., 2016). We report on the BLEU (Papineni et al., 2002) and Perplexity (PPL) metric to provide a comparison with the current state-of-the-art methods. We also report our performance using other metrics such as length of responses produced by the model. Following, Mei *et al* (2017), we also report the diversity metric that helps us measure the ability of the model to promote diversity in responses (Li et al.,

[1]More information about the dataset made available on the (Rashkin et al., 2019)

[2]`https://github.com/huggingface/pytorch-transformers`

2016a). Diversity is calculated as the as the number of distinct unigrams in the generation scaled by the total number of generated tokens (Mei et al., 2017; Li et al., 2016c). We report on two additional automated metrics of readability and coherence. Readability quantifies the linguistic quality of text and the difficulty of the reader in understanding the text (Novikova et al., 2017). We measure readability through the Flesch Reading Ease (FRE) (Kincaid et al., 1975) which computes the number of words, syllables and sentences in the text. Higher readability scores indicate that utterance is easier to read and comprehend. Similarly, coherence measures the ability of the dialogue system to produce responses consistent with the topic of conversation. To calculate coherence, we use the method proposed by Dziri *et al.* (2018).

4 Results

4.1 Automated Metrics

We first compare the performance of our approach with the baseline results obtained from Rashkin *et al.* (2019) that uses a full transformer architecture (Vaswani et al., 2017), consisting of an encoder and decoder. Table 3 provides a comparison of our approach with to the baseline approach. In Table 3, we refer our "*Our Model Fine-Tuned*" as the baseline fine-tuned GPT-2 model trained on the dialogue and "*Our-model Emo-prepend*" as the GPT-2 model that is fine-tuned on the dialogues but also conditioned on the emotion displayed in the conversation. We find that fine-tuning the GPT-2 language model using a transfer learning approach helps us achieve a lower perplexity and a higher BLEU scores. The results from our approach are consistent with the empirical study conducted by Edunov *et al* (2019) that demonstrate the effectiveness of the using pre-trained model diminishes when added to the decoder network in an *seq2seq* approach. We also perform a comparison between our two models on the metrics of length, diversity, readability and coherence. We find that our baseline model produces less diverse responses compared to when the model is conditioned on emotion. We find that the our *emo-prepend* model also higher a slightly higher readability score that our baseline model.

4.2 Qualitative Evaluation

To assess the quality of generations, we conducted a MTurk human evaluation. We recruited a total of 15 participants and each participant was asked to evaluate 25 randomly sampled outputs from the test set on three metrics:

1. Readability - Is the response easy to understand, fluent and grammatical and does not have any consecutive repeating words.
2. Coherence - Is the response relevant to the context of the conversation.
3. Emotional Appropriateness- Does the response convey emotion suitable to the context of the conversation?

Table 5 shows the results obtained from the human evaluation comparing the performance of our fine-tuned, emotion pre-pend model to the ground-truth response. We find that our fine-tuned model outperforms the emo-prepend on all three metrics from the ratings provided by the human ratings.

5 Related Work

The area of dialogue systems has been studied extensively in both open-domain (Niu and Bansal, 2018) and goal-oriented (Lipton et al., 2018) situations. Extant approaches towards building dialogue systems has been done predominantly through the *seq2seq* framework (Vinyals and Le, 2015). However, prior research has shown that these systems are prone to producing dull and generic responses that causes engagement with the human to be affected (Vinyals and Le, 2015; Venkatesh et al., 2018). Researchers have tackled this problem of dull and generic responses through different optimization function such as MMI (Li et al., 2016b) and through reinforcement learning approaches(Li et al., 2016d). Alternative approaches towards generating more engaging responses is by grounding them in personality of the speakers that enables in creating more personalized and consistent responses (Li et al., 2016c; Zhang et al., 2018; Wolf et al., 2019).

Several other works have focused on creating more engaging responses by producing affective responses. One of the earlier works to incorporate affect through language modeling is the work done by Ghosh *et al.* (Ghosh et al., 2017). This work leverages the LIWC (Pennebaker et al., 2001) text analysis platform for affective features. Alternative approaches of inducing emotion in generated responses from a *seq2seq* framework include the work done by Zhou *et al*(2018) that uses internal and external memory, Asghar *et al.* (2018) that models emotion through af-

Experiment	Valid PPL	BLEU	Readability	Coherence	Length	Diversity
Baseline Fine-Tuned (Rashkin et al., 2019)	21.24	6.27	x	x	x	x
Baseline Emo-prepend (Rashkin et al., 2019)	24.30	4.36	x	x	x	x
Our Model Fine-Tuned	**18.32**	7.71	0.78	0.93	9.77	0.0031
Our Model Emo-prepend	19.49	**7.78**	**0.79**	0.93	9.71	**0.0033**

Table 3: Comparison of the performance of our model to the baseline model proposed by Rashkin *et al* (2019) across a variety of automated metrics to provide a thorough comparison. **x** indicates that these metrics were not provided in the Rashkin *et al* (2019) work.

Input Conversation	Model	Outputs
Person A: I think I aced my supervisor exam!	Ground Truth	Woo hoo! That is awesome! Congrats! When will you find out
	Fine Tuned	What was it?
	Emo Prepend	Wow, that's impressive!
Person A: I had pizza at a restaurant a couple days ago and found a hair on my pizza!	Ground Truth	It is disgusting. What did you do?
	Fine Tuned	I was so angry
	Emo Prepend	That's terrible, did you clean it up?

Table 4: Example generations from our two model along with the ground truth responses.

	Readability	Coherence	Emotional Appropriate-ness
Our Model Fine-Tuned	4.14	3.50	3.70
Our Model Emo-prepend	3.54	3.4	3.19
Ground Truth	3.92	3.86	4

Table 5: Human ratings demonstrating a comparison between our models to the ground truth responses on the metrics of readability, coherence and emotional appropriateness

fective embeddings and Huang *et al* (2018) that induce emotion through concatenation with input sequence. More recently, introduction of transformer based approaches have helped advance the state of art across several natural language understanding tasks (Vaswani et al., 2017). These transformers models have also helped created large pre-trained language models such as BERT (Devlin et al., 2019), XL-NET (Yang et al., 2019), GPT-2 (Radford et al., 2019). However, these pre-trained models show inconsistent behavior towards language generation (Ziegler et al., 2019).

6 Conclusion and Discussion

In this work, we study how pre-trained language models can be adopted for conditional language generation on smaller datasets. Specifically, we look at conditioning the pre-trained model on the emotion of the situation produce more affective responses that are appropriate for a particular situation. We notice that our fine-tuned and emo-prepend models outperform the current state of the art approach relative to the automated metrics such as BLEU and perplexity on the validation set. We also notice that the emo-prepend approach does not out perform a simple fine tuning approach on the dataset. We plan to investigate the cause of this in future work from the perspective of better experiment design for evaluation (Santhanam and Shaikh, 2019) and analyzing the models focus when emotion is prepended to the sequence (Clark et al., 2019). Along with this, we also notice other drawbacks in our work such as not having an emotional classifier to predict the outcome of the generated sentence, which we plan to address in future work.

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No FA8650-18-C-7881. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of AFRL, DARPA, or the U.S. Government. We thank the anonymous reviewers for the helpful feedback.

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.

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*, pages 65–72.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

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.

Nouha Dziri, Ehsan Kamalloo, Kory W Mathewson, and Osmar Zaiane. 2018. Augmenting neural response generation with context-aware topical attention. *arXiv preprint arXiv:1811.01063*.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. 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 4052–4059, Minneapolis, Minnesota. Association for Computational Linguistics.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-LM: A neural language model for customizable affective text generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 634–642, Vancouver, Canada. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Chenyang Huang, Osmar Zaiane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. 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 49–54.

Daphne Ippolito, Reno Kriz, Joao Sedoc, Maria Kustikova, and Chris Callison-Burch. 2019. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3752–3762, Florence, Italy. Association for Computational Linguistics.

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.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016b. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016c. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003, Berlin, Germany. Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016d. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2018. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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 *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132. Association for Computational Linguistics.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2017. Coherent dialogue with attention-based language models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, San Francisco, CA.

Tong Niu and Mohit Bansal. 2018. Polite dialogue generation without parallel data. *Transactions of the Association of Computational Linguistics*, 6:373–389.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. 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.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, 71(2001):2001.

Soujanya Poria, Navonil Majumder, Rada Mihalcea, and Eduard Hovy. 2019. Emotion recognition in conversation: Research challenges, datasets, and recent advances. *arXiv preprint arXiv:1905.02947*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: a new benchmark and dataset. In *ACL*.

Sashank Santhanam and Samira Shaikh. 2019. Towards best experiment design for evaluating dialogue system output. *arXiv preprint arXiv:1909.10122*.

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)*, pages 1715–1725, Berlin, Germany. 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.

Anu Venkatesh, Chandra Khatri, Ashwin Ram, Fenfei Guo, Raefer Gabriel, Ashish Nagar, Rohit Prasad, Ming Cheng, Behnam Hedayatnia, Angeliki Metallinou, et al. 2018. On evaluating and comparing conversational agents. *arXiv preprint arXiv:1801.03625*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *arXiv preprint arXiv:1901.08149*.

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*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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*.

Noun Generation for Nominalization in Academic Writing

Dariush Saberi, John Lee
Department of Linguistics and Translation
City University of Hong Kong
dsaberi2-c@my.cityu.edu.hk, jsylee@cityu.edu.hk

Abstract

Nominalization is a common technique in academic writing for producing abstract and formal text. Since it often involves paraphrasing a clause with a verb or adjectival phrase into a noun phrase, an important task is to generate the noun to replace the original verb or adjective. Given that a verb or adjective may have multiple nominalized forms with similar meaning, the system needs to be able to automatically select the most appropriate one. We propose an unsupervised algorithm that makes the selection with BERT, a state-of-the-art neural language model. Experimental results show that it significantly outperforms baselines based on word frequencies, word2vec and doc2vec.

1 Introduction

Automatic paraphrasing — re-writing a sentence while preserving its original meaning — has received much interest in the computational linguistics community in recent years. One type of paraphrasing is lexical substitution (McCarthy and Navigli, 2009), which replaces a word or short phrase with another. Paraphrasing can also involve manipulation of the clausal structure of a sentence, with a range of options that has been described as the "cline of metaphoricity" (Halliday and Matthiessen, 2014). Towards one end of this cline, the text offers a "congruent construal of experience", and the sentences tend to be clausally complex but lexically simple (e.g., the complex clause "Because she didn't know the rules, she died"[1]). Towards the other end of the cline, the text exhibits a "metaphorical reconstrual", and the sentences are clausally simpler and lexically denser (e.g., the nominal group "Her death through ignorance of the rules").

Previous studies on automatic manipulation of clausal structure have mostly concentrated on syntactic simplification, typically by splitting a complex sentence into two or more simple sentences (Siddharthan, 2002; Aluísio et al., 2008; Narayan and Gardent, 2014). More recent research has also attempted semi-automatic nominalization (Lee et al., 2018), which aims to paraphrase a complex clause into a simplex clause by transforming verb or adjectival phrases into noun phrases.

Noun generation is a core task in the nominalization pipeline (Table 2). Resources such as NOMLEX (Meyers et al., 1998) and CATVAR (Habash and Dorr, 2003) have greatly facilitated this task by providing lists of related nouns, verbs and adjectives. However, straightforward look-up in these lists does not suffice since a word may have multiple nominalized forms with similar meaning. For example, the verb "dominate" can be transformed into "domination", "dominance", "dominion", as well as the gerund form "dominating". We will henceforth refer to these as the "***noun candidates***". As shown in Table 1, in the context of the clause "The British dominated India", "domination" would be preferred (i.e., "British *domination* of India"); in the context of the clause "older people dominated this neighborhood", "dominance" would be more appropriate (i.e., "The *dominance* of older people in this neighborhood").

The goal of this paper is to evaluate a noun generation algorithm that selects the best noun candidate during nominalization. The approach taken by Lee et al. (2018), which considers noun frequency statistics alone, always selects the same noun regardless of the sentential context. We use instead a neural language model, BERT, for noun generation. Experimental results show that it significantly outperforms baselines based on word

[1]This example and the next are both taken from Halliday and Matthiessen (2014).

Verb-to-noun mapping	Example sentence	Nominalized version
dominate → {dominance, domination, ...}	The British **dominated** India ... Older people **dominated** this neighborhood ...	British **domination** of India ... The **dominance** of older people in this neighborhood ...
move → {motion, move, ...}	They **moved** northward ... The particle **moved** irregularly ...	Their **move** northward ... The irregular **motion** of the particle ...
enter → {entrance, entry, ...}	The clown **entered** the stage ... The immigrants **entered** the country ...	The clown's **entrance** to the stage ... The **entry** of the immigrants into the country ...
measure → {measure, measurement, ...}	Success is **measured** ... Blood pressure is **measured** ...	The **measure** of success ... The **measurement** of blood pressure ...

Table 1: Example verb-to-noun mappings with multiple *noun candidates* (left column), illustrated by sentences with the same verb (middle column) requiring different *target nouns* (right column) in their nominalized version.

frequencies, word2vec and doc2vec.

The rest of the paper is organized as follows. Following a review of previous work (Section 2), we give details on our dataset (Section 3) and outline our approach (Section 4). We then report experimental results (Section 5) and conclude.

2 Previous work

We first discuss the relation between our task and lexical substitution (Section 2.1) and word sense disambiguation (Section 2.2). We then describe an existing nominalization system (Section 2.3), whose noun generation algorithm will serve as our baseline.

2.1 Relation to lexical substitution

Noun generation in nominalization can be considered a specialized kind of lexical substitution. While lexical substitution typically aims for a paraphrase in the same part-of-speech (POS) (e.g., "dominate" → "prevail"), our task by definition involves a change in POS, usually from a verb or adjective to a noun (e.g., "dominate" → "domination"). This difference is reflected in the limited number of verb-noun or adjective-noun entries in open-source paraphrase corpora such as PPDB (Ganitkevitch et al., 2013).

2.2 Relation to word sense disambiguation

Word sense disambiguation (WSD) is relevant to noun generation to the extent that verb senses can guide the choice of noun candidates. For example, "succeed" in the sense of "achieve the desired result" should be paraphrased as "success" ("He succeeded in ..." → "His success in ..."), whereas "succeed" in the sense of "take over a position" would require "succession" ("He succeeded to the throne ..." → "His succession to the throne ...").

WSD is not necessary for noun generation when the verb corresponds to a noun with the same range of meanings. Consider the verb "conclude", which may mean either "to finish" or "to reach agreement". Nominalization requires no WSD since the noun "conclusion" preserves the same semantic ambiguity.

In other cases, our task requires fine-grained WSD, especially when the noun candidates are semantically close. Their differences can be rather nuanced (e.g., "domination" vs. "dominance"), making it challenging for typical WSD models to distinguish.

2.3 Nominalization pipeline

In the first reported tool for semi-automatic nominalization aimed at academic writing (Lee et al., 2018), the system first parses the input clause to detect the potential for nominalization. If its dependency tree exhibits an expected structure (e.g., Table 2(i)), the system proceeds to lexical mapping (Table 2(ii)), which includes transforming the main verb ("entered") to a noun ("entrance"); an adverb ("abruptly") to an adjective ("abrupt"); and the subject ("the clown") to a possessive form ("the clown's" or "of the clown") . Finally, the system generates a sentence by choosing one of the possible surface realizations through heuristics (Table 2(iii)).

The noun generation task in lexical mapping utilizes verb-to-noun and adjective-to-noun mappings, some examples of which are shown in Table 1. The system constructed these mappings on the basis of NOMLEX (Meyers et al., 1998) and

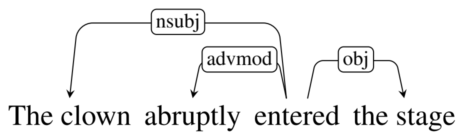
(i) Parsing	 nsubj advmod obj The clown abruptly entered the stage				
(ii) Lexical mapping	the clown	abruptly	**entered**	the stage	
	↓	↓	↓	↓	
	the clown’s	abrupt	**entrance**	to the stage	
(iii) Sentence generation	The	abrupt	entrance	of the clown	to the stage ...
	The clown’s	abrupt	entrance		to the stage ...
	His	abrupt	entrance		to the stage ...

Table 2: The nominalization pipeline (Lee et al., 2018): (i) syntactic parsing; (ii) lexical mapping, including noun generation (bolded), which is the focus of this paper; and (iii) sentence generation.

CATVAR (Habash and Dorr, 2003)[2], with a total of 7,879 verb-to-noun mappings, and 11,369 adjective-noun mappings.

3 Dataset

Among the mappings described in Section 2.3, there were 7,380 verb-to-noun and 5,339 adjective-to-noun mappings with at least two noun candidates. We constructed our dataset on the basis of these mappings only, because the others do not require selection from multiple candidates.

The ideal dataset for this research would consist of input sentences containing these verbs and adjectives; and, as gold output, the noun candidate selected for use in the nominalized version of these sentences. Unfortunately, no such large-scale dataset exists. One option is to sample sentences in a corpus and ask human experts to nominalize them; this would however require considerable manual annotation. To avoid this cost, an alternative is to work backwards: identify sentences containing noun phrases that could plausibly be the result of nominalization (e.g., those in the right column of Table 1). This methodology produces the gold noun candidate automatically. One can then retrieve from the mappings the verb or adjective that would be in the hypothetical sentence before nominalization (e.g., those in the middle column of Table 1). Adopting this methodology, we constructed a challenging dataset by prioritizing verbs and adjectives that are more ambiguous, i.e., those with more noun candidates.

One potential issue is the plausibility of the selected sentences as the nominalized form of an input sentence. To make our dataset as realistic as possible, we required sentences to have one of the three common nominalized forms, corresponding to the three surface forms shown in Table 2(iii):

- “the <target noun> of <subject> ...”
- “<subject>’s <target noun> ...”
- “<poss> <target noun> ...”

where <target noun> is the gold noun candidate, <poss> is a possessive pronoun and <subject> is the noun subject of the hypothetical input sentence before nominalization. In addition, we require the target noun, verb and adjective to be tagged as such at least two times in the Brown Corpus (Francis and Kučera, 1979), to avoid words with rare usage.

Our dataset consists of a total of 620 sentences that satisfy the above requirements, including 332 retrieved from the Brown Corpus and 288 from the British Academic Written English (BAWE) Corpus (Nesi, 2008). The sentences contain 73 distinct verbs and 19 distinct adjectives, each with an average of 2.67 noun candidates.

4 Approach

The noun generation algorithm used by Lee et al. (2018) considers only the word frequency statistics of the noun candidates. It therefore always chooses the same noun candidate for a verb (or adjective), even if the sentential context warrants a different choice due to word sense, register or fluency considerations.

To remove this limitation, we use BERT (Devlin et al., 2019), a state-of-the-art neural language model based on the “Transformer” architec-

[2]Verbs-to-be and modal verbs were not treated.

ture (Vaswani et al., 2017). BERT has been shown to be effective in a wide range of natural language processing tasks. The model is bi-directional, i.e., trained to predict the identity of a masked word based on the words both before and after it. We consider the suitability of each noun candidate in the verb-to-noun and adjective-to-noun mappings as the masked word.

In each sentence in our dataset, we mask the target noun and ask BERT for its word predictions for the masked position.[3] Among the noun candidates, we identify the highest-ranked one among the first 15,000 word predictions. If none of the candidates is ranked, we create a sentence with each candidate by replacing the masked word with it, and obtain the BERT score for the sentence. We select the candidate that yields the sentence with the highest score.

5 Results

We compared our proposed approach with four baselines:

Spelling This baseline selects the noun candidate that has the smallest letter edit distance from the original verb or adjective.

Frequency Following Lee et al. (2018), this baseline selects the noun candidate with the highest unigram frequency count in the *Google Web 1T Corpus* (Brants and Franz, 2006).

Word2vec We select the noun candidate that is most similar to the original verb or adjective, as estimated by the Google News pre-trained Gensim model (Mikolov et al., 2013).

Doc2vec We select the noun candidate that has the highest cosine similarity with the sentence embeddings, taking each sentence as a small "document".[4]

As shown in Table 3, the Frequency baseline achieved higher accuracy than the Spelling baseline and Word2vec. The frequency of a noun candidate appears to serve as a good proxy for its appropriateness. All three approaches, however, ignore the specific context of the sentence, always proposing the same noun for a given verb or adjective.

Approach	Brown	BAWE
Frequency	53.92%	48.61%
Spelling	46.39%	35.07%
Word2vec	35.84%	43.71%
Doc2vec	36.74%	38.88%
BERT	**74.10%**	**72.57%**

Table 3: Accuracy of our proposed noun generation algorithm with BERT, compared to baselines.

By taking the rest of the sentence into account when predicting the noun candidate, BERT yielded better performance. Consider the verb "measure". Although frequency favors the noun "measure", BERT was able to select "measurement" when it collocates with "quantity". While Doc2vec also considers the sentential context, it did not perform as well as BERT, likely because the masked language modeling objective offers a better fit for our task.

Still, BERT's performance was limited by difficulties in recognizing nuanced differences between noun pairs such as "use" and "usage", or "occupation" and "occupancy". With access only to a single sentence, it was also unable to choose formal words such as "continuance" over "continuation" when called for by the context.

6 Conclusion

We propose an unsupervised algorithm for noun generation from a verb or adjectival phrase, a task that is essential for automatic nominalization system for academic writing. This algorithm selects the most appropriate noun candidate with BERT, a state-of-the-art neural language model. Experimental results show that it significantly outperforms baselines based on word frequencies, word2vec and doc2vec.

Acknowledgments

This work was partially funded by a HKSAR UGC Teaching Learning Grant (Meeting the Challenge of Teaching and Learning Language in the University: Enhancing Linguistic Competence and Performance in English and Chinese) in the 2016-19 Triennium; and by an Applied Research Grant (#9667151) from City University of Hong Kong.

[3]We used the PyTorch implementation of BERT with the bert-base-uncased model.

[4]We used the following settings: max epocs = 100, vector size = 20, alpha = 0.025, min count = 1, dm = 1. With word embeddings combined, the best results were obtained with dbow = 0 and dmpv = 0

References

Sandra Aluísio, Lucia Specia, T. A. Pardo, E. G. Maziero, and R. P. Fortes. 2008. Towards Brazilian Portuguese Automatic Text Simplification Systems. In *Proc. 8th ACM Symposium on Document Engineering*.

Thorsten Brants and Alex Franz. 2006. The Google Web 1T 5-gram Corpus Version 1.1. In *LDC2006T13*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pretraining of Deep Bidirectional Transformers for Language Understanding. In *Proc. NAACL-HLT*.

W. N. Francis and H. Kučera. 1979. Manual of Information to Accompany a Standard Corpus of Present-Day Edited American English, for use with Digital Computers. Providence, RI. Department of Linguistics, Brown University.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proc. NAACL-HLT*.

Nizar Habash and Bonnie Dorr. 2003. A Categorial Variation Database for English. In *Proc. NAACL*.

M. A. K. Halliday and C. M. I. M. Matthiessen. 2014. *Halliday's Introduction to Functional Grammar*. Routledge.

John Lee, Dariush Saberi, Marvin Lam, and Jonathan Webster. 2018. Assisted Nominalization for Academic English Writing. In *Proc. Workshop on Intelligent Interactive Systems and Language Generation (2ISNLG)*, pages 26–30.

Diana McCarthy and Roberto Navigli. 2009. The English Lexical Substitution Task. *Language Resources and Evaluation*, 43:139–159.

Adam Meyers, Catherine Macleod, Roman Yangarber, Ralph Grishman, Leslie Barrett, and Ruth Reeves. 1998. Using NOMLEX to Produce Nominalization Patterns for Information Extraction. In *Proc. Computational Treatment of Nominals*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proc. International Conference on Learning Representations (ICLR)*.

Shashi Narayan and Claire Gardent. 2014. Hybrid Simplification using Deep Semantics and Machine Translation. In *Proc. ACL*.

Hilary Nesi. 2008. BAWE: an introduction to a new resource. In *Proc. Eighth Teaching and Language Corpora Conference*, page 239–46, Lisbon, Portugal. ISLA.

Advaith Siddharthan. 2002. An Architecture for a Text Simplification System. In *Proc. Language Engineering Conference (LEC)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *Advances in Neural Information Processing Systems*, pages 6000–6010.

Efficient text generation of user-defined topic using generative adversarial networks

Chenhan Yuan
Dept. of Computer Science
Virginia Tech
VA, USA
chenhan@vt.edu

Yi-chin Huang
Dept. of Computer Science
National Pingtung University
Pingtung, Taiwan
ychin.huang@gmail.com

Cheng-Hung Tsai
Institute for Information Industry
Taipei, Taiwan
jasontsai@iii.org.tw

Abstract

This study focused on efficient text generation using generative adversarial networks (GAN). Assuming that the goal is to generate a paragraph of a user-defined topic and sentimental tendency, conventionally the whole network has to be re-trained to obtain new results each time when a user changes the topic. This would be time-consuming and impractical. Therefore, we propose a User-Defined GAN (UD-GAN) with two-level discriminators to solve this problem. The first discriminator aims to guide the generator to learn paragraph-level information and sentence syntactic structure, which is constructed by multiple-LSTMs. The second one copes with higher level information, such as the user-defined sentiment and topic for text generation. The cosine similarity based on TF-IDF and length penalty are adopted to determine the relevance of the topic. Then, the second discriminator is re-trained with generator if the topic or sentiment for text generation is modified. The system evaluations are conducted to compare the performance of the proposed method with other GAN-based ones. The objective results showed that the proposed method is capable of generating texts with less time than others and the generated text are related to the user-defined topic and sentiment. We will further investigate the possibility of incorporating more detailed paragraph information such as semantics into text generation to enhance the result.

1 Introduction

Text generation, as a basic natural language processing task, has many applications, such as dialogue robots (Li et al., 2017), machine translation (Hu et al., 2017), paraphrasing (Power and Scott, 2005) and so on. With the rise of deep learning, different neural networks are introduced to generate text. For example, researchers use the recurrent neural network (RNN) (Mikolov et al., 2010) to train the language model because of its capability to process sequential data. However, the RNN suffers from the gradient vanishing problem (Hochreiter, 1998) when the sequence becomes longer. To address this problem, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997)is further adopted as a sequential neural network model to generate sentences.

Lately, the Generative Adversarial Networks (GAN) framework (Goodfellow et al., 2014) has been introduced into the NLP community. GAN has two different models for completing the data-generating task. One of them is Generator G, which is responsible for generating data, and another one is discriminator D, which determines whether the input data is the real data or not. The generator G continuously optimizes generated data based on the judgment of discriminator D. After several epochs, the generated data will become more realistic.

However, GAN was originally designed to process continuous data, and using discrete data as input would make it impossible to update the gradients of the GAN framework(Huszár, 2015). To process discrete data, several variants of the GAN model for generating text have been proposed. These GAN variants could achieve good performances in text generation task, such as MaskGAN (Fedus et al., 2018), RankGAN (Lin et al., 2017), and TextGAN (Zhang et al., 2016).

In order to make these models fit the distribution of real text data better, the number of parameters of text generation models based on neural network are increased, which means that training these neural network models often takes a lot of time even using GPU. Conventionally, topic-related text generation models incorporate an arbitrary topic as an input by adopting mechanisms like attention (Feng et al., 2018). Therefore, each time when the user wants to generate new sentences with another topic or sentimental tendency, the text generation models have to be retrained with all parameters to satisfy new requirements. In some scenarios, e.g., news generation, spending lots of time retraining model is not practical and the user wants new responding quickly.

To tackle this problem, a novel text generation model based on GAN is proposed, which is called User-Defined Generative Adversarial Networks (UD-GAN). The key idea is to separate the sentence syntax model as the basic model and the topic-related model as a higher-level model, and these two could be trained independently from each other. So, when the topic or other user-defined information is modified, e.g., sentimental tendency, only one of both models needs to be retrained. In this way, once the basic syntax model is established, the following training will become much faster, since only the higher-level model has to be retrained.

In our proposed method, the discriminator is constructed based on this idea. One of the discriminators called discriminator-general, which learns to determine the proper context information and whether the input sentence is a valid syntactic structure. Another discriminator is called the discriminator-special, which ensures the output is user-defined. Inspired by SeqGAN (Yu et al., 2017), we use the evaluation results of the generated text from discriminators as a reward to guide the generator to select future actions, which is to generate an updated word.

For training the discriminator-special, it will take feature vectors as input, instead of sentences. The feature vector is defined based on the sentiment detection and topic relevance of generated sentence. The cosine similarity based on TF-IDF and length penalty are jointly adopted to represent topic relevance.

Note that the UD-GAN is designed to be more practical to generate short paragraphs, which means sentences generated by it should be context-aware and behave like a paragraph together with surrounding sentences. To achieve this idea, discriminator-general is designed with hierarchical multiple LSTM layers. The LSTM at the top of the network processes paragraph-level information while the bottom LSTMs process sentence-level information.

The organization of the paper is as follows: First, we discussed the related works of our method in the section 2. The proposed method is described in the Section 3, including the feature extraction and model definition and training. In the Section 4, the experiment settings and evaluation results of the comparing methods are depicted. Finally, the concluding remarks and future works are described in the Section 5.

Algorithm 1 Initial training generator $G\theta$, discriminator-special $D\gamma$, discriminator-general $D\phi$

```
1:  Initialize $G\theta$, $D\phi$ and $D\gamma$ with random weights$\theta$, $\phi$ and $\gamma$
2:  Pre-train $G\theta$ using MLE on real text data set
3:  Generate negative samples using $G\theta$ to train $D\phi$ and $D\gamma$
4:  Generate synthetic positive samples to train $D\gamma$
5:  Minimizing the cross entropy to pre-train $D\gamma$
6:  Minimizing the cross entropy to pre-train$D\phi$
7:  for $i \leftarrow 1$ to $M$ do
8:      for $j \leftarrow 1$ to $N$ do
9:          Generate a sequence $Y_{1:T} \backsim G\theta$
10:         Compute rewards via Eq.5
11:         Update parameters of $G\theta$ via Eq.4
12:     end for
13:     for $k \leftarrow 1$ to $P$ do
14:         Generate negative samples using $G\theta$
15:         Train $D\phi$ with negative samples and real text data via Eq.6
16:     end for
17:     for $l \leftarrow 1$ to $T$ do
18:         Generate feature vectors corresponding to negative samples generated by $G\theta$
19:         Generate synthetic feature vectors
20:         Train $D\gamma$ with negative and synthetic feature vectors via Eq.6
21:     end for
22: end for
```

2 Related Work

Text generation is a basic task in natural language processing (NLP). In previous works, many researchers (Power and Scott, 2005) extracted grammar rules from text to generate new texts. These works are capable of generating semantically rich and grammatically correct text, but due to the fixed grammar rules, generated sentences are quite lack of diversity. As neural networks could fit the dis-

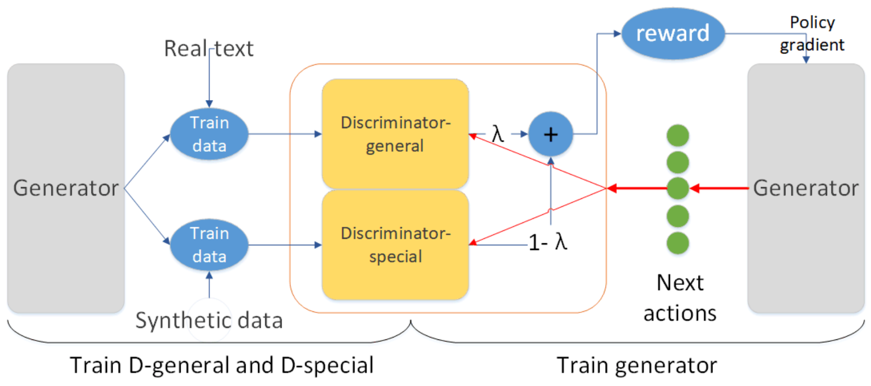


Figure 1: The framework of the proposed UD-GAN

Algorithm 2 Following training generator $G\theta$, discriminator-special $D\gamma$

```
1:  Initialize $G\theta$, $D\gamma$ with random weights$\theta$, $D\gamma$
2:  Load trained $D\phi$
3:  Do 2∽5 steps in Algorithm 1
4:  **for** $i \leftarrow 1$ to $M$ **do**
5:      Do 8∽12 steps in Algorithm 1
6:      **for** $l \leftarrow 1$ to $T$ **do**
7:          Generate feature vectors corresponding
            to negative samples generated by $G\theta$
8:          Generate synthetic feature vectors
9:          Train $D\gamma$ with negative and synthetic
            feature vectors via Eq.6
10:     **end for**
11: **end for**
```

tribution of real data better, some researchers design GAN-based models as language models to generate text. Unlike standard GAN, the loss function or training method of generator are modified to enable GAN to process discrete data.

For example, In TextGAN (Zhang et al., 2016), researchers apply feature matching with standard GAN loss function to train the generator. Reinforcement learning (Sutton et al., 2000) is another useful machine learning technique to train model with unlabeled data. Trained model will choose next actions to maximize expected reward, which is given by interface environment. Yu proposed SeqGAN (Yu et al., 2017), which combine reinforcement learning with GAN. In SeqGAN, the generator uses the result of discriminator as a reward and choose next actions, which is to generate the next words in text generation task. To generate longer text, LeakGAN (Guo et al., 2018) is introduced to enable the discriminator leaks features extracted from its input to generator, which then uses this signal to guide the outputs in each generation step before generating the entire sentence.

Another vital application of NLP is the sentiment analysis (Pang et al., 2008; Wilson et al., 2005). Generally, the sentiment analysis task measures the emotional tendency of the whole sentence based on the word usage that can represent emotions in that sentence. Therefore, the establishment of an emotional word dictionary is essential. Affective Norms for English Words (ANEW) (Bradley and Lang, 1999) lexicon sorts all words according to rating score from 1 to 9. The highest score means the sentence convey a very positive emotion, and the lowest one represents the most negative emotion for the sentence. Based on that, some researchers (Hutto and Gilbert, 2014) construct a gold-standard list of lexical features then combine these lexical features with consideration for five general rules, which could represent the sentiment of a sentence. The VADER algorithm proposes a rule-based sentiment analyzer that has outperformed the other machine learning-based algorithms.

3 Proposed Method

3.1 Basic Structure of UD-GAN

As shown in Fig.1, UD-GAN contains a generator G_θ that is capable of generating context-dependent sentences and the two-level discriminators. Discriminator-general D_ϕ guides the generator to learn the paragraph-level information and correct syntactic structure, while discriminator-special D_γ determines whether the generated text is related to the user-defined topic and sentiment. Discriminator-special D_γ is trained with synthetic perfect data and generated text data, while discriminator-general D_ϕ is trained with real text data and generated text data.

As we apply reinforcement learning with policy gradient to train the generator, the outputs of the two discriminators for the generated text will be combined and served as a reward to train the generator. Generator G_θ will choose the best next actions based on the reward it received. After the first training via Algorithm 1, the discriminator-general parameters are saved as the pre-trained model. In the subsequent trainings, we only train the parameters of the generator G_θ and discriminator-special D_γ via Algorithm 2. The details about training method and structure of discriminators and generator are described as follows.

3.2 The Framework of D-Special

The Feature Vector of D-Special

Discriminator-special D_γ takes a vector containing 5 elements as input, which could represent the sentimental and topical relevance of each sentence.

In our model, users can describe the cause and effect of an event in one sentence, which is used as the topic for generating sentences. We use the first element to represent the similarity between sentence entered by the user and generated sentence, which could also represent the user-defined topic relevance of the generated text. Based on the TF-IDF (Sparck Jones, 1972) value of each word in the sentence that the user entered and the generated sentence, the cosine similarity between these two sentences is calculated as a parameter to measure the user-defined topic relevance of the generated sentence. A larger value of cosine similarity means that the generated sentence is related to the user-defined topic.

However, if only this element is used to instruct the generator G_θ to generate topic-related sentences, the resulting sentences will be substantially as long as the user-defined topic sentence. More importantly, the generated sentences will lack diversity with same meaning. Therefore, we propose the second element, length penalty, to reduce the negative impact of the first element. The difference between the length of the generated sentence and the length of the topic sentence defined by user is mapped in [0, 1] via Eq.1.

$$penalty_{g'} = \frac{|len_{g'} - len_i|}{\max_{g\in G}|len_g - len_i| - \min_{g\in G}|len_g - len_i|} \quad (1)$$

where i is input sentence, g' is the evaluated generated sentence and G is the set of generated sentences. We set 0.5 to the optimal length penalty, which means that if the length of the sentence is very close to or very far from the length of topic sentence, it is unqualified.

We implemented the VADER algorithm to calculate the probability that a generated sentence belongs to a positive, negative or neutral emotion class. As VADER gives three values that correspond to the probability of each sentiment category, the sum of which is 1, these three values will be saved in the third to fifth elements. The optimal sentiment is defined by the user.

In conventional GAN training, the discriminator treats real text data as the positive sample and generated text as the negative sample. However, there is no sentence in real corpus that has exactly the same features as the positive sample, since its feature vector is constructed by applying the above mention algorithm, while the user-defined feature vector is a specific value. Therefore, we train the discriminator-special D_γ with synthetic data, which is treated as positive sample. For example, supposing that the user would like to generate an essay with one positive emotion, then the UD-GAN will generate [1,0.5,1,0,0] vectors corresponding to the number of generated sentences, which will be combined with vectors corresponding to the generated sentences as the input of discriminator-special.

The Structure of D-Special

Two linear layers with Relu as the activation function are used as discriminator-special D_γ. The output of this network will be part of the reward to train generator G_θ after it passed through a softmax layer.

We explain here why the multiple fully connected layer is implemented as a discriminator-special. The first reason is that after the Discriminator-General is constructed, in the subsequent training, the discriminator-special will be continuously retrained when demands of user change. This requires spending as little time as possible to train a good discriminator-special. The

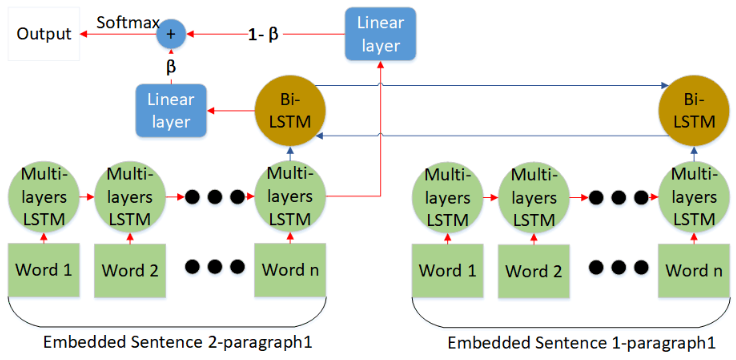


Figure 2: The proposed framework for Discriminator-general

multiple fully connected layer has fewer parameters, which means this network will converge faster than others will. Another reason is that the aim of training discriminator-special is to distinguish whether the input vector corresponds the user-defined one. For an input with only five variables, a neural network with two fully connected layers is complicated enough to determine the class of input vector correctly.

3.3 The Framework of D-General

Unlike conventional ideas of using classifier-based models as a discriminator, the discriminator-general D_ϕ needs to process sequence data and context information, such as the paragraph information for each sentence to generate paragraph-level text.

Therefore, as shown in Fig.2, we designed a hierarchical-multiple-LSTM neural network as the discriminator-general D_ϕ. The bottom multi-layers LSTM takes an embedding vector for each word in a sentence as the input and it outputs a feature matrix representing the corresponding sentence. The top bidirectional LSTM (Graves and Schmidhuber, 2005) takes the feature matrices of these sentences, which belong to the same paragraph, as input and it outputs a feature matrix representing that paragraph. After transforming through two different linear layers respectively, the above two feature matrices will be combined together. Finally, the discriminator-general calculates the score of the input sentence via Eq. 2.

$$R(Y) = softmax[(1-\beta)LSTM_\alpha + \beta LSTM_\eta] \tag{2}$$

where β is a trainable parameter ranging 0-1.

3.4 Generator

Generator G_θ is designed with GRU (Chung et al., 2014). In UD-GAN, due to the excessive parameters of the two discriminators, it is easy to guide the generator to be over-fitting. As a commonly used variant of LSTM, GRU avoids this over-fitting problem. In addition, having fewer parameters than conventional LSTM allows GRU to take less time to converge, which is the first priority in UD-GAN.

3.5 Reward and Policy Gradient Training

The reinforcement learning has been incorporated to enable GAN to process discrete data. In this scenario, generator G_θ will use the results from discriminators on the generated text as reward to generate next words. In UD-GAN, the reward is calculated based on results of two discriminators. Generator G_θ tries to maximize expected reward from the initial state till the end state via Eq.3(loss function).

$$\begin{aligned} J(\theta) &= \sum_{t=1}^{T} \mathbb{E}(R_t|S_{t-1},\theta) \\ &= \sum_{t=1}^{T} G_\theta(y_t|Y)[\lambda(D_\phi(Y)) + (1-\lambda)D_\gamma(Y))] \end{aligned} \tag{3}$$

Where λ is a manually set weight and Y is a complete sequence and R_t is the reward for a whole sequence. In our experiments, we set λ to 0.8 to give more weight to the discriminator-general D_ϕ for generating sentences with better syntactic structure. Note that since discriminators can only make the judgement with a complete sequence, the Monte Carlo search (Silver et al., 2016) is adopted to find out some of the possible generated complete sequences of each state. the average judgment results of the discriminators for these sequences are calculated as a reward of this state.

In this paper, we implemented policy gradient method. The gradient of Eq.3 can be derived approximately as follows:

$$\nabla_\theta J(\theta) \simeq \sum_{t=1}^{T} \mathbb{E}_{y_t \sim G_\theta}[\nabla_\theta log G_\theta(y_t|Y) Q^{G_\theta}_{D_\phi,D_\gamma}(y_t|Y)] \quad (4)$$

where $Q^{G_\theta}_{D_\phi,D_\gamma}(y_t|y_{1:t-1})$ can be derived via Eq.5.

$$Q^{G_\theta}_{D_\phi,D_\gamma}(y_t|y_{1:t-1}) = \lambda(D_\phi(Y)) + (1-\lambda)D_\gamma(Y)) \quad (5)$$

The loss function of both discriminators is introduced as follows:

$$J = -(\mathbb{E}_{Y \backsim P_{data}}[R(Y)] - \mathbb{E}_{Y \backsim G_\theta}[1 - R(Y)]) \quad (6)$$

where $R(Y)$ is the reward from two discriminators for a whole sequence.

4 Experimental Analysis

4.1 Dataset

We crawled nearly 10,000 press released from the opinion section of Newsweek as the training corpus. The opinion section of Newsweek is selected as training corpus because the paragraphs of the essays in Newsweek are generally closely related and not long. The other reason is that through the articles in the opinion section, authors can often convey their own sentiment tendencies.

NER is used to replace name-entities with their name-entity tags to decrease vocabulary. After tokenizing the corpus, long sentences of more than 45 words in the corpus were removed. The final training corpus has 425K sentences and 103K paragraphs.

4.2 Experimental Setting

SeqGAN and LeakGAN are used as the baseline system to evaluate UD-GAN. We train SeqGAN and LeakGAN for 20 epochs, which is same as the number of times UD-GAN is trained. Other parameters of baselines remain unchanged as implemented in their original papers.

The bottom of the discriminator-general consists of three layers of LSTM. The hidden dimension of discriminator-general bidirectional LSTMs for the UD-GAN and the bottom LSTMs is set to 64. Besides, the hidden dimension of discriminator-special linear layer and GRU unit of generator is set to 32. In each epoch of initial training, generator G is trained once, and the discriminator-general is trained four times while the discriminator-special is trained twice.

For evaluating the effectiveness of the proposed method, we first compared the sentences relevance to user-defined topic and sentimental tendency, and then compare the training time of each system. Finally, the fluency and correctness of UD-GAN and baseliens were evaluated.

GAN-based models	ROUGE-L
UD-GAN(GS)	364.73
UD-GAN(S)	**370.54**
UD-GAN(G)	340.19
SeqGAN	342.27
LeakGAN	345.03

Table 1: The ROUGE-L score for each system. UD-GAN(G+S) represents initial training and UD-GAN(S) represents following training. UD-GAN(G) only has discriminator-general and generator. Note that this score is the sum of all generated sentences' ROUGE-L results.

4.3 Relevance of Topic and Sentiment

Relevance of Topic

As an objective summary accuracy evaluation method that is widely used, ROUGE (Lin, 2004) is also adopted here to evaluate whether generated sentences are related to user-defined topics. Generated sentences are treated as summaries to be evaluated, and the topic sentence defined by user is used as a reference summary to evaluate whether the generated sentence is related to the topic. Note that even if the ROUGE scores of the generated sentences are not high, it does not mean that these sentences are not closely related to the user-defined topic necessarily. One possibility is that the generated sentences will use other words or syntactic structures to describe the topic sentence.

In this paper, we report the sum of ROUGE-L scores of all sentences. Based on the longest common subsequence, ROUGE-L is a score related to recall rate. As shown in Table.1, the ROUGE-L scores for UD-GAN (G+S) and UD-GAN(S) are slightly higher than baseline systems and UD-GAN (G).

	Positive	Negative	Neutral
UD-GAN(GS)	0.39	0.05	0.56
UD-GAN(S)	**0.41**	**0.04**	**0.55**
UD-GAN(G)	0.10	0.08	0.82
SeqGAN	0.09	0.08	0.83
LeakGAN	0.08	0.07	0.85

Table 2: The probability of sentiment tendency of generated sentences

Relevance of Sentimental Tendency

The VADER algorithm is used to calculate the probability that the sentimental tendency of the generated sentences to be positive, negative or neutral. Here, we evaluated the system performance by setting the target sentimental tendency as positive.

As shown in Table.2, the average probability in each sentimental tendency category of all sentences is calculated. With training discriminator-special, UD-GAN (G+S) and UD-GAN (S) are more likely to generate positive sentences than baselines. Which proves that the proposed method is capable to generate the sentences with the desired sentiment. However, since the total number of sentences expressing positive sentimental tendency in the training corpus is quite low, the probability of UD-GAN generating positive sentiment is still not higher than 0.5.

Generate Context-dependent Sentences

To demonstrate that UD-GAN can generate context-dependent sentences, we show sentences generated by UD-GAN and baselines. As shown in Table 3, one can see that the proposed UD-GAN does generate sentences related to the user-defined topic. UD-GAN tries to add some conjunctions when generating sentences so that the sentences seem to be related, and each sentence is extended with other related words based on the topic. Note that there are some Name-Entity (NE) tags generated by the models because the NE tagging has been done for simplifying the corpus lexicon.

However, semantically, these sentences are not intrinsically related to each other, which is a problem we will address in the future.

4.4 Training Time Evaluation

The time spending on gradient propagation and update of UD-GAN and baselines are compared, instead of the time spending on loading and saving data. Our platform is a workstation with a GeForce

topic: the attack in douma occurred days after trump indicated that he wanted to pull us troops out
UD-GAN(S):
1. the country contacts to the u.s. and trains **troops** for government living on the federal system in LOCATION .
2. we are discussed actively : if u.s. is the facts that citizens in the country will likely vote for type elections ?
3. during these **attack** things **occurred days** , i say just PERSON who **pulls** in the exchange best **troops out** as trade in LOCATION .
4. and he often enthusiastic , telling only having heard nothing happened while you can indicate to **pull out** from country .
5. but these generations in LOCATION can predict the next five **attacks occur**.
LeakGAN:
1. it prompted the opposition during a “ real ” of subtlety , and video straws .
2. but if PERSON know that we serve the best drives these country purposes . ”
3. besides disarming our administration and pricing and its traditional views .
4. with her contempt for all enough neighbors . ”
5. one day i ’d go beyond my candor .
SeqGAN:
1. we do n’t mean .
2. you should be “ changed ” that you know .
3. i ’ve always been proposing the findings .
4. in other words , he ’s because you have a testament to his goodness – not a result .
5. he gave economic law .

Table 3: An example of the generated sentences from different systems

GAN-based models	Time s
UD-GAN(GS)	29061.48
UD-GAN(S)	**4841.99**
UD-GAN(G)	29036.65
SeqGAN	27011.08
LeakGAN	30471.95

Table 4: Time spending on training of each models

GTX 1080 Ti graphics card with 11G RAM. All GAN-based models compared here are implemented in pytorch (Paszke et al., 2017) framework to eliminate the impact of different frameworks on time consumption.

As shown in Table.4, because the structure of discriminator-general is more complex than the structure of discriminator D of baselines, initial training of UD-GAN takes the longest time. However, in the subsquent trainings, due to the gradient propagation and parameter update of discriminator-special is quite fast, the time required to train UD-GAN (S) is the shortest. The UD-GAN (S) takes only about an hour and a half to complete training, which is much less than the nearly eight hours of training time for baselines.

4.5 Fluency and Accuracy

As shown in table 5, we report BLEU (Papineni et al., 2002) scores of UD-GAN and baselines to compare the fluency and accuracy of text they generate. The BLEU we use here is the average value of 1-gram BLEU, 2-gram BLEU and 3-gram BLEU, which are given the same weights .

In the case of training the discriminator-general only, the BLEU score of the UD-GAN (G) is between SeqGAN and LeakGAN. Therefore, the accuracy and fluency evaluation of using multi-layer LSTMs as a discriminator is comparable to that of using a classifier-based model, such as CNN, as the discriminator. When the discriminator-general and discriminator-special are simultaneously trained (initial training), UD-GAN (G+S) has a slightly higher BLEU score than UD-GAN (G). That is to say, even if discriminator-special is added and the result of discriminator-general, which can distinguish the correctness of the sentence, is less weighted, the resultant generator of UD-GAN (G+S) can still learn how to generate a sentence with the correct syntax. Then we change the user-defined topic and sentimental tendency to train the discriminator-special only (subsequent training). The results showed that the BLEU score of the UD-GAN(S) is still between LeakGAN and SeqGAN. It means that retraining the discriminator-special has no effect on whether the generator can learn the correct syntax without changing the weights of rewards generated by discriminator-general and discriminator-special.

GAN-based models	BLEU score
UD-GAN(G+S)	0.6412
UD-GAN(S)	0.6409
UD-GAN(G)	0.6357
SeqGAN	0.6303
LeakGAN	**0.7161**

Table 5: The average BLEU score for each system. Note that UD-GAN(S) achieves comparable BLEU performance with baselines, whose training needs far less time than baselines.

5 Conclusion and Future Work

In this paper, we propose a UD-GAN method to re-train text generation model more efficiently to generate sentences that are consistent with the new user-defined topic and sentimental tendency. We compared the accuracy and fluency of sentences generated by UD-GAN with other GAN-based text generation models. The experimental results showed that sentences generated by UD-GAN are competent. Meanwhile, UD-GAN takes much less time in the re-train stage than other models. According to experimental results, UD-GAN can also successfully generate sentences related to the user-defined topic and sentimental tendency, while baselines does not have this capability. Besides, UD-GAN can also generate paragraph-level text.

However, the sentences generated by UD-GAN are still inferior to the state-of-the-art method, i.e., LeakGAN, in terms of fluency. And the current paragraph-level information used here does not include complex linguistic information, such as the order of sentences. In future work, we will try to maintain the existing advantages of UD-GAN while improving the readability of generated text.

References

Margaret M Bradley and Peter J Lang. 1999. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Citeseer.

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*.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the_. *arXiv preprint arXiv:1801.07736*.

Xiaocheng Feng, Ming Liu, Jiahao Liu, Bing Qin, Yibo Sun, and Ting Liu. 2018. Topic-to-essay generation with neural networks. In *IJCAI*, pages 4078–4084.

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.

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

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

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. Controllable text generation. *arXiv preprint arXiv:1703.00955*, 7.

Ferenc Huszár. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, pages 3155–3165.

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*.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

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.

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.

Richard Power and Donia Scott. 2005. Automatic generation of large-scale paraphrases.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*.

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*.

Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating text via adversarial training. In *NIPS workshop on Adversarial Training*, volume 21.