

# Lexicon Infused Phrase Embeddings for Named Entity Resolution

Alexandre Passos, Vineet Kumar, Andrew McCallum

School of Computer Science

University of Massachusetts, Amherst

{apassos, vineet, mccallum}@cs.umass.edu

## Abstract

Most state-of-the-art approaches for named-entity recognition (NER) use semi supervised information in the form of word clusters and lexicons. Recently neural network-based language models have been explored, as they as a byproduct generate highly informative vector representations for words, known as word embeddings. In this paper we present two contributions: a new form of learning word embeddings that can leverage information from relevant lexicons to improve the representations, and the first system to use neural word embeddings to achieve state-of-the-art results on named-entity recognition in both CoNLL and Ontonotes NER. Our system achieves an F1 score of 90.90 on the test set for CoNLL 2003—significantly better than any previous system trained on public data, and matching a system employing massive private industrial query-log data.

## 1 Introduction

In many natural language processing tasks, such as named-entity recognition or coreference resolution, syntax alone is not enough to build a high performance system; some external source of information is required. In most state-of-the-art systems for named-entity recognition (NER) this knowledge comes in two forms: domain-specific lexicons (lists of word types related to the desired named entity types) and word representations (either clusterings or vectorial representations of word types which capture some of their syntactic and semantic behavior and allow generalizing to unseen word types).

Current state-of-the-art named entity recognition systems use Brown clusters as the form of word representation (Ratinov and Roth, 2009; Turian et al., 2010; Miller et al., 2004; Brown et al., 1992), or other cluster-based representations computed from private data (Lin and Wu, 2009). While very attractive due to their simplicity, generality, and hierarchical structure, Brown clusters are limited because the computational complexity of fitting a model scales quadratically with the number of words in the corpus, or the number of “base clusters” in some efficient implementations, making it infeasible to train it on large corpora or with millions of word types.

Although some attempts have been made to train named-entity recognition systems with other forms of word representations, most notably those obtained from training neural language models (Turian et al., 2010; Collobert and Weston, 2008), these systems have historically underperformed simple applications of Brown clusters. A disadvantage of neural language models is that, while they are inherently more scalable than Brown clusters, training large neural networks is still often expensive; for example, Turian et al (2010) report that some models took multiple days or weeks to produce acceptable representations. Moreover, language embeddings learned from neural networks tend to behave in a “nonlinear” fashion, as they are trained to encourage a many-layered neural network to assign high probability to the data. These neural networks can detect nonlinear relationships between the embeddings, which is not possible in a log-linear model such as a conditional random field, and therefore limiting how much information from the embeddings can be actually leveraged.

Recently Mikolov et al (Mikolov et al., 2013a;

Mikolov et al., 2013b) proposed two simple log-linear language models, the CBOW model and the Skip-Gram model, that are simplifications of neural language models, and which can be very efficiently trained on large amounts of data. For example it is possible to train a Skip-gram model over more than a billion tokens with a single machine in less than half a day. These embeddings can also be trained on phrases instead of individual word types, allowing for fine granularity of meaning.

In this paper we make the following contributions. (1) We show how to extend the Skip-Gram language model by injecting supervisory training signal from a collection of curated lexicons—effectively encouraging training to learn similar embeddings for phrases which occur in the same lexicons. (2) We demonstrate that this method outperforms a simple application of the Skip-Gram model on the semantic similarity task on which it was originally tested. (3) We show that a linear-chain CRF is able to successfully use these log-linearly-trained embeddings better than the other neural-network-trained embeddings. (4) We show that lexicon-infused embeddings let us easily build a new highest-performing named entity recognition system on CoNLL 2003 data (Tjong Kim Sang and De Meulder, 2003) which is trained using only publicly available data. (5) We also present results on the relatively understudied Ontonotes NER task (Weischedel et al., 2011), where we show that our embeddings outperform Brown clusters.

## 2 Background and Related Work

### 2.1 Language models and word embeddings

A statistical language model is a way to assign probabilities to all possible documents in a given language. Most such models can be classified in one of two categories: they can directly assign probabilities to sequences of word types, such as is done in  $n$ -gram models, or they can operate in a lower-dimensional latent space, to which word types are mapped. While most state-of-the-art language models are  $n$ -gram models, the representations used in models of the latter category, henceforth referred to as “embeddings,” have been found to be useful in many NLP applications which don’t actually need a language model. The underlying intuition is that when language models compress the information about the word types in

a latent space they capture much of the commonalities and differences between word types. Hence features extracted from these models then can generalize better than features derived from the word types themselves.

One simple language model that discovers useful embeddings is known as *Brown clustering* (Brown et al., 1992). A Brown clustering is a class-based bigram model in which (1) the probability of a document is the product of the probabilities of its bigrams, (2) the probability of each bigram is the product of the probability of a bigram model over latent classes and the probability of each class generating the actual word types in the bigram, and (3) each word type has non-zero probability only on a single class. Given a one-to-one assignment of word types to classes, then, and a corpus of text, it is easy to estimate these probabilities with maximum likelihood by counting the frequencies of the different class bigrams and the frequencies of word tokens of each type in the corpus. The Brown clustering algorithm works by starting with an initial assignment of word types to classes (which is usually either one unique class per type or a small number of seed classes corresponding to the most frequent types in the corpus), and then iteratively selecting the pair of classes to merge that would lead to the highest post-merge log-likelihood, doing so until all classes have been merged. This process produces a hierarchical clustering of the word types in the corpus, and these clusterings have been found useful in many applications (Ratinov and Roth, 2009; Koo et al., 2008; Miller et al., 2004). There are other similar models of distributional clustering of English words which can be similarly effective (Pereira et al., 1993).

One limitation of Brown clusters is their computational complexity, as training takes  $O(kV^2 + N)x$  time to train, where  $k$  is the number of base clusters,  $V$  size of vocabulary, and  $N$  number of tokens. This is infeasible for large corpora with millions of word types.

Another family of language models that produces embeddings is the *neural language models*. Neural language models generally work by mapping each word type to a vector in a low-dimensional vector space and assigning probabilities to  $n$ -grams by processing their embeddings in a neural network. Many different neural language models have been proposed (Bengio et al., 2003; Morin and Bengio, 2005; Bengio, 2008;



detecting named entities is superficially simple, since most sequences of capitalized words are named entities (excluding headlines, sentence beginnings, and a few other exceptions), finding all entities is non trivial, and determining the correct named entity type can sometimes be surprisingly hard. Performing the task well often requires external knowledge of some form.

In this paper we evaluate our system on two labeled datasets for NER: CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) and Ontonotes (Weischedel et al., 2011). The CoNLL dataset has approximately 320k tokens, divided into 220k tokens for training, 55k tokens for development, and 50k tokens for testing. While the training and development sets are quite similar, the test set is substantially different, and performance on it depends strongly on how much external knowledge the systems have. The CoNLL dataset has four entity types: PERSON, LOCATION, ORGANIZATION, AND MISCELLANEOUS. The Ontonotes dataset is substantially larger: it has 1.6M tokens total, with 1.4M for training, 100K for development, and 130k for testing. It also has eighteen entity types, a much larger set than the CoNLL dataset, including works of art, dates, cardinal numbers, languages, and events.

The performance of NER systems is commonly measured in terms of precision, recall, and F1 on the sets of entities in the ground truth and returned by the system.

### 2.3.1 Baseline System

In this section we describe in detail the baseline NER system we use. It is inspired by the system described in Ratnoff and Roth (2009).

Because NER annotations are commonly not nested (for example, in the text “the US Army”, “US Army” is treated as a single entity, instead of the location “US” and the organization “US Army”) it is possible to treat NER as a sequence labeling problem, where each token in the sentence receives a label which depends on which entity type it belongs to and its position in the entity. Following Ratnoff and Roth (2009) we use the BILOU encoding, where each token can either BEGIN an entity, be INSIDE an entity, be the LAST token in an entity, be OUTSIDE an entity, or be the single UNIQUE token in an entity.

Our baseline architecture is a stacked linear-chain CRF (Lafferty et al., 2001) system: we train two CRFs, where the second CRF can condition

on the predictions made by the first CRF as well as features of the data. Both CRFs, following Zhang and Johnson (2003), have roughly similar features.

While local features capture a lot of the clues used in text to highlight named entities, they cannot necessarily disambiguate entity types or detect named entities in special positions, such as the first tokens in a sentence. To solve these problems most NER systems incorporate some form of external knowledge. In our baseline system we use lexicons of months, days, person names, companies, job titles, places, events, organizations, books, films, and some minor others. These lexicons were gathered from US Census data, Wikipedia category pages, and Wikipedia redirects (and will be made publicly available upon publication).

Following Ratnoff and Roth (2009), we also compare the performance of our system with a system using features based on the Brown clusters of the word types in a document. Since, as seen in section 2.1, Brown clusters are hierarchical, we use features corresponding to prefixes of the path from the root to the leaf for each word type.

More specifically, the feature templates of the baseline system are as follows. First for each token we compute:

- its word type;
- word type, after excluding digits and lower-casing it;
- its capitalization pattern;
- whether it is punctuation;
- 4-character prefixes and suffixes;
- character  $n$ -grams from length 2 to 5;
- whether it is in a wikipedia-extracted lexicon of person names (first, last, and honorifics), dates (months, years), place names (country, US state, city, place suffixes, general location words), organizations, and man-made things;
- whether it is a demonym.

For each token’s label we have feature templates considering all token’s features, all neighboring token’s features (up to distance 2), and bags of words of features of tokens in a window of size 8 around each token. We also add a feature marking whether a token is the first occurrence of its word type in a document.

When using Brown clusters we add as token features all prefixes of lengths 4, 6, 10, and 20, of its brown cluster.

For the second-layer model we use all these features, as well as the label predicted for each token

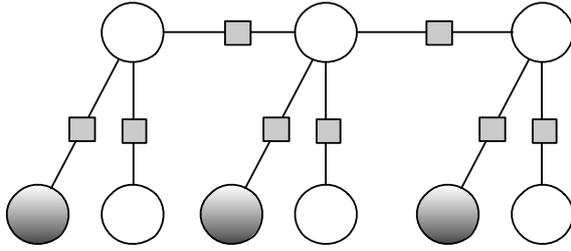


Figure 2: Chain CRF model for a NER system with three tokens. Filled rectangles represent factors. Circles at top represent labels, circles at bottom represent binary token based features. Filled circles indicate the phrase embeddings for each token.

by the first-layer model.

As seen in the Experiments Section, our baseline system is competitive with state-of-the-art systems which use similar forms of information.

We train this system with stochastic gradient ascent, using the AdaGrad RDA algorithm (Duchi et al., 2011), with both  $\ell_1$  and  $\ell_2$  regularization, automatically tuned for each experimental setting by measuring performance on the development set.

## 2.4 NER with Phrase Embeddings

In this section we describe how to extend our baseline NER system to use word embeddings as features.

First we group the tokens into phrases, assigning to each token a single phrase greedily. We prefer shorter phrases over longer ones, since our embeddings are often more reliable for the shorter phrases, and since the longer phrases in our dictionary are mostly extracted from Wikipedia page titles, which are not always semantically meaningful when seen in free text. We then add factors connecting each token’s label with the embedding for its phrase.

Figure 2 shows how phrase embeddings are plugged into a chain-CRF based NER system. Following Turian (2010), we scale the embedding vector by a real number, which is a hyperparameter tuned on the development data. Connecting tokens to phrase embeddings of their neighboring tokens did not improve performance for phrase embeddings, but it was mildly beneficial for token embeddings.

## 3 Lexicon-infused Skip-gram Models

The Skip-gram model as defined in Section 2.2 is fundamentally trained in unsupervised fashion using simply words and their  $n$ -gram contexts. Injecting some NER-specific supervision into the embeddings can make them more relevant to the NER task.

Lexicons are a simple yet powerful way to provide task-specific supervisory information to the model without the burden of labeling additional data. However, while lexicons have proven useful in various NLP tasks, a small amount of noise in a lexicon can severely impair its usefulness as a feature in log-linear models. For example, even legitimate data, such as the Chinese last name “He” occurring in a lexicon of person last names, can cause the lexicon feature to fire spuriously for many training tokens that are labeled PERSON, and then this lexicon feature may be given low or even negative weight.

We propose to address both these problems by employing lexicons as part of the word embedding training. The skip-gram model can be trained to predict not only neighboring words but also lexicon membership of the central word (or phrase). The resulting embedding training will thus be somewhat supervised by tending to bring together the vectors of words sharing a lexicon membership. Furthermore, this type of training can effectively “clean” the influence of noisy lexicons because even if “He” appears in the PERSON lexicon, it will have a sufficiently different context distribution than labeled named person entities (*e.g.* a lack of preceding honorifics, etc) that the presence of this noise in the lexicon will not be as problematic as it was previously.

Furthermore, while Skip-gram models can be trained on billions of tokens to learn word embeddings for over a million word types in a single day, this might not be enough data to capture reliable embeddings of all relevant named entity phrases. Certain sets of word types, such as names of famous scientists, can occur infrequently enough that the Skip-gram model will not have enough contextual examples to learn embeddings that highlight their relevant similarities.

In this section we describe how to extend the Skip-gram model to incorporate auxiliary information from lexicons, or lists of related words, encouraging the model to assign similar embeddings to word types in similar lexicons.

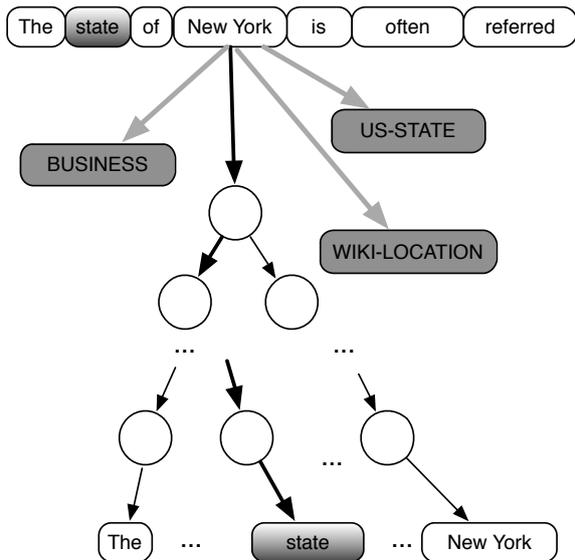


Figure 3: A Semi supervised Skip-gram Model. “New York” predicts the word “state”. With lexicon-infusion, “New York” also predicts its lexicon classes: US-State, Wiki-location

In the basic Skip-gram model, as seen in Section 2.2, the likelihood is, for each n-gram, a product of the probability of the embedding associated with the middle word conditioned on each context word. We can inject supervision in this model by also predicting, given the embedding of the middle word, whether it is a member of each lexicon. Figure 3 shows an example, where the word “New York” predicts “state”, and also its lexicon classes: Business, US-State and Wiki-Location.

Hence, with subscript  $s$  iterating over each lexicon (or set of related words), and  $l_s^{m_n}$  being a label for whether each word is in the set, and  $w_s$  indicating the parameters of its classifier, the full likelihood of the model is

$$\prod_{n \in N} \left( \prod_{c_n \in n} \prod_i \sigma(l_i^{c_n} w_i^{c_n T} e_{m_n}) \right) \left( \prod_s \sigma(l_s^{m_n} w_s^T e_{m_n}) \right). \quad (2)$$

This is a simple modification to equation (1) that also predicts the lexicon memberships. Note that the parameters  $w_s$  of the auxiliary per-lexicon classifiers are also learned. The lexicons are not inserted in the binary tree with the words; instead, each lexicon gets its own binary classifier.

**Algorithm 1** Generating the training examples for lexicon-infused embeddings

---

```

1: for all  $n$ -gram  $n$  with middle word  $m_n$  do
2:   for all Context-word  $c_n$  do
3:     for all Classifier, label pair  $(w_i^{c_n}, l_i^{c_n})$ 
       in the tree do
4:       Add training example
        $e_{m_n}, w_i^{c_n}, l_i^{c_n}$ 
5:     end for
6:   end for
7:   for all Lexicon  $s$ , with label  $l_s^{m_n}$  do
8:     Add training example  $e_{m_n}, w_s, l_s^{m_n}$ 
9:   end for
10: end for

```

---

In practice, a very small fraction of words are present in a lexicon-class and this creates skewed training data, with overwhelmingly many negative examples. We address this issue by aggressively sub-sampling negative training data for each lexicon class. We do so by randomly selecting only 1% of the possible negative lexicons for each token.

A Skip-gram model has  $V$  binary classifiers. A lexicon-infused Skip-gram model predicts an additional  $K$  classes, and thus has  $V + K$  binary classifiers. If number of classes  $K$  is large, we can induce a tree over the classes, similarly to what is done over words in the vocabulary. In our trained models, however, we have one million words in the vocabulary and twenty-two lexicons, so this is not necessary.

## 4 Experiments

Our phrase embeddings are learned on the combination of English Wikipedia and the RCV1 Corpus (Lewis et al., 2004). Wikipedia contains 8M articles, and RCV1 contains 946K. To get candidate phrases we first select bigrams which have a pointwise mutual information score larger than 1000. We discard bigrams with stopwords from a manually selected list. If two bigrams share a token we add its corresponding trigram to our phrase list. We further add page titles from the English Wikipedia to the list of candidate phrases, as well as all word types. We get a total of about 10M phrases. We restrict the vocabulary to the most frequent 1M phrases. All our reported experiments are on 50-dimensional embeddings. Longer embeddings, while performing better on the semantic similarity task, as seen in Mikolov et al (2013a;

Model	Accuracy
Skip-Gram	29.89
Lex-0.05	30.37
Lex-0.01	<b>30.72</b>

Table 1: Accuracy for Semantic-Syntactic task, when restricted to Top 30K words. Lex-0.01 refers to a model trained with lexicons, where 0.01% of negative examples were used for training.

2013b), did not perform as well on NER.

To train phrase embeddings, we use a context of length 21. We use lexicons derived from Wikipedia categories and data from the US Census, totaling  $K = 22$  lexicon classes. We use a randomly selected 0.01% of negative training examples for lexicons.

We perform two sets of experiments. First, we validate our lexicon-infused phrase embeddings on a semantic similarity task, similar to Mikolov et al (Mikolov et al., 2013a). Then we evaluate their utility on two named-entity recognition tasks.

For the NER Experiments, we use the baseline system as described in Section 2.3.1. NER systems marked as ‘‘Skip-gram’’ consider phrase embeddings; ‘‘LexEmb’’ consider lexicon-infused embeddings; ‘‘Brown’’ use Brown clusters, and ‘‘Gaz’’ use our lexicons as features.

#### 4.1 Syntactic and Semantic Similarity

Mikolov et al. (2013a) introduce a test set to measure syntactic and semantic regularities for words. This set contains 8869 semantic and 10675 syntactic questions. Each question consists of four words, such as big, biggest, small, smallest. It asks questions of the form ‘‘What is the word that is similar to *small* in the same sense as *biggest* is similar to *big*’’. To test this, we compute the vector  $X = vector(‘‘biggest’’) - vector(‘‘big’’) + vector(‘‘small’’)$ . Next, we search for the word closest to  $X$  in terms of cosine distance (excluding ‘‘biggest’’, ‘‘small’’, and ‘‘big’’). This question is considered correctly answered only if the closest word found is ‘‘smallest’’. As in Mikolov et al (Mikolov et al., 2013a), we only search over words which are among the 30K most frequent words in the vocabulary.

Table 1 depicts the accuracy on Semantic Syntactic Task for models trained with 50 dimensions. We find that lexicon-infused embeddings perform better than Skip-gram. Further, lex-0.01 performs

System	Dev	Test
Baseline	92.22	87.93
Baseline + Brown	93.39	90.05
Baseline + Skip-gram	93.68	89.68
Baseline + LexEmb	93.81	89.56
Baseline + Gaz	93.69	89.27
Baseline + Gaz + Brown	93.88	90.67
Baseline + Gaz + Skip-gram	94.23	90.33
Baseline + Gaz + LexEmb	<b>94.46</b>	<b>90.90</b>
Ando and Zhang (2005)	93.15	89.31
Suzuki and Isozaki (2008)	<b>94.48</b>	89.92
Ratinov and Roth (2009)	93.50	90.57
Lin and Wu (2009)	-	<b>90.90</b>

Table 2: Final NER F1 scores for the CoNLL 2003 shared task. On the top are the systems presented in this paper, and on the bottom we have baseline systems. The best results within each area are highlighted in bold. Lin and Wu 2009 use massive private industrial query-log data in training.

the best, and we use this model for further NER experiments. There was no perceptible difference in computation cost from learning lexicon-infused embeddings versus learning standard Skip-gram embeddings.

#### 4.2 CoNLL 2003 NER

We applied our models on CoNLL 2003 NER data set. All hyperparameters were tuned by training on training set, and evaluating on the development set. Then the best hyperparameter values were trained on the combination of training and development data and applied on the test set, to obtain the final results.

Table 2 shows the phrase F1 scores of all systems we implemented, as well as state-of-the-art results from the literature. Note that using traditional unsupervised Skip-gram embeddings is worse than Brown clusters. In contrast, our lexicon-infused phrase embeddings **Lex-0.01** achieves 90.90—a state-of-the-art F1 score for the test set. This result matches the highest F1 previously reported, in Lin and Wu (2009), but is the first system to do so without using massive private data. Our result is significantly better than the previous best using public data.

#### 4.3 Ontonotes 5.0 NER

Similarly to the CoNLL NER setup, we tuned the hyperparameters on the development set. We use

System	Dev	Test
Baseline	79.04	79.85
Baseline + Brown	79.95	81.38
Baseline + Skip-gram	80.59	81.91
Baseline + LexEmb	80.65	81.82
Baseline + Gaz	79.85	81.31
Baseline + Gaz + Brown	80.53	82.05
Baseline + Gaz + Skip-gram	80.70	<b>82.30</b>
Baseline + Gaz + LexEmb	<b>80.81</b>	82.24

Table 3: Final NER F1 scores for Ontonotes 5.0 dataset. The results in bold face are the best on each evaluation set.

the same list of lexicons as for CoNLL NER.

Table 3 summarize our results. We found that both Skip-gram and Lexicon infused embeddings give better results than using Brown Clusters as features. However, in this case Skip-gram embeddings give marginally better results. (So as not to jeopardize our ability to fairly do further research on this task, we did not analyze the test set errors that may explain this.) These are, to the best of our knowledge, the first published performance numbers on the Ontonotes NER task.

## 5 Conclusions

We have shown how to inject external supervision to a Skip-gram model to learn better phrase embeddings. We demonstrate the quality of phrase embeddings on three tasks: Syntactic-semantic similarity, CoNLL 2003 NER, and Ontonotes 5.0 NER. In the process, we provide a new public state-of-the-art NER system for the widely contested CoNLL 2003 shared task.

We demonstrate how we can plug phrase embeddings into an existing log-linear CRF System.

This work demonstrates that it is possible to learn high-quality phrase embeddings and fine-tune them with external supervision from billions of tokens within one day computation time. We further demonstrate that learning embeddings is important and key to improve NLP Tasks such as NER.

In future, we want to explore employing embeddings to other NLP tasks such as dependency parsing and coreference resolution. We also want to explore improving embeddings using error gradients from NER.

## References

- Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 1–9. Association for Computational Linguistics.
- Yoshua Bengio, Rejean Ducharme, and Pascal Vincent. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.
- Yoshua Bengio. 2008. Neural net language models. *Scholarpedia*, 3(1):3881.
- Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.
- Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.
- Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. 2011. Multi-view learning of word embeddings via cca. In *Advances in Neural Information Processing Systems*, pages 199–207.
- Paramveer Dhillon, Jordan Rodu, Dean Foster, and Lyle Ungar. 2012. Two step cca: A new spectral method for estimating vector models of words. *arXiv preprint arXiv:1206.6403*.
- John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 999999:2121–2159.
- David A Huffman. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.
- Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing.
- John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
- David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397.
- Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics.

- Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.
- Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT-NAACL*, volume 4, pages 337–342. Citeseer.
- Andriy Mnih and Geoffrey E Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.
- Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252.
- Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 183–190. Association for Computational Linguistics.
- Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.
- Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. In *ACL*, pages 665–673. Citeseer.
- Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.
- Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.
- Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2011. *OntoNotes Release 4.0*. Linguistic Data Consortium.
- Tong Zhang and David Johnson. 2003. A robust risk minimization based named entity recognition system. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 204–207. Association for Computational Linguistics.