A preliminary evaluation of word representations for named-entity recognition

Joseph Turian
Département d'Informatique et
Recherche Opérationnelle (DIRO)
Université de Montréal
Montréal, Québec, Canada, H3T 1J4
lastname@iro.umontreal.ca

Lev Ratinov
Department of Computer Science

University of Illinois at Urbana-Champaign
201 N. Goodwin Avenue, Urbana, IL 61801
ratinov2@uiuc.edu

Yoshua Bengio
Département d'Informatique et
Recherche Opérationnelle (DIRO)
Université de Montréal
Montréal, Québec, Canada, H3T 1J4
bengioy@iro.umontreal.ca

Dan Roth
Department of Computer Science

University of Illinois at Urbana-Champaign
201 N. Goodwin Avenue, Urbana, IL 61801
danr@uiuc.edu

Abstract

We use different word representations as word features for a named-entity recognition (NER) system with a linear model. This work is part of a larger empirical survey, evaluating different word representations on different NLP tasks. We evaluate Brown clusters, Collobert and Weston (2008) embeddings, and HLBL (Mnih & Hinton, 2009) embeddings of words. All three representations improve accuracy on NER, with the Brown clusters providing a larger improvement than the two embeddings, and the HLBL embeddings more than the Collobert and Weston (2008) embeddings. We also discuss some of the practical issues in using embeddings as features. Brown clusters are simpler than embeddings because they require less hyperparameter tuning.

1 Introduction

Lexicalized NLP models require word features to be provided in some input representation. Conventionally, we take the word and convert it to a symbolic ID, which we then transform to a feature vector using a *one-hot* representation: The feature vector has the same length as the size of the vocabulary, and only one dimension is on. However, the one-hot representation of a word suffers from data sparsity: Namely, for words that are rare in the labeled training data, their corresponding model parameters will be poorly estimated. Moreover, at test time, the model cannot handle out-of-vocabulary words (OOV, words that do not appear in the labeled training data). Unlike one-hot representations of words, distributed representations have the potential of generalizing naturally to sequences semantically similar to those seen during training (Bengio et al., 2001; Bengio, 2008), These limitations of one-hot word representations have prompted researchers to investigate unsupervised methods for inducing word representations over large unlabeled corpora.

One common approach to inducing unsupervised word representation is to use clustering, perhaps hierarchical, which were used by a variety of researchers (Miller et al., 2004; Liang, 2005; Koo et al., 2008; Ratinov & Roth, 2009; Huang & Yates, 2009). This leads to a one-hot representation over a smaller vocabulary size, which have the disadvantage of losing much information about the original word. Neural language models (Bengio et al., 2001; Schwenk & Gauvain, 2002; Mnih & Hinton,

2007; Collobert & Weston, 2008), on the other hand, induce dense real-valued low-dimensional word embeddings using unsupervised approaches. (See Bengio (2008) for a more complete list of references on neural language models.)

Unsupervised word representations have been used in previous NLP work, and have demonstrated improvements in generalization accuracy on a variety of tasks. But different word representations have never been systematically compared in a controlled way. In this work, we compare different techniques for inducing word representations, evaluating them on the task of named entity recognition (NER)—identifying people, organizations, locations and other named entities in text. In particular, we focus on the applicability of distributed word representations as inputs to linear classifiers.

This work is part of a larger empirical survey, evaluating different word representations on different NLP tasks using near state-of-the-art systems. Our goal is to distribute word features for off-the-shelf use in all NLP tasks.

2 Clustering-based word representations

One type of word representation is to induce a clustering over words.

2.1 Hard clustering

The Brown algorithm is a hierarchical clustering algorithm which clusters words to maximize the mutual information of bigrams (Brown et al., 1992). So it is a class-based bigram language model. It runs in time $O(V \cdot K^2)$, where V is the size of the vocabulary and K is the number of clusters.

The hierarchical nature of the clustering means that we can choose the word class at several levels in the hierarchy, which can compensate for poor clusters of a small number of words. One downside of this approach is that it is based solely on bigram statistics, and does not consider word usage in a wider context.

Brown clusters have been used successfully in a variety of NLP applications: NER (Miller et al., 2004; Liang, 2005; Ratinov & Roth, 2009), PCFG parsing (Candito & Crabbé, 2009), dependency parsing (Koo et al., 2008), and semantic dependency parsing (Zhao et al., 2009).

Lin and Wu (2009) present a non-hierarchical clustering algorithm for phrases, which uses MapReduce. They achieve state-of-the-art accuracy on NER using their phrase clusters. Martin et al. (1998) presents algorithms for inducing hierarchical clusterings based upon word bigram *and trigram* statistics. Ushioda (1996) presents an extension to the Brown clustering algorithm, and learn hierarchical clusterings of word as well as phrases, which they apply to POS tagging.

2.2 Soft clustering

HMMs can be used to induce a soft clustering, specifically a multinomial distribution over possible clusters (hidden states). Unlike all previous approaches, which assign a specific word representation to each word *type*, HMM word representations are assigned to word *tokens*. The benefit of HMM representations is that they can model polysemy and other context-specific characteristics of each word token. On the downside, it is more difficult to distribute and use the HMM representations than representations which are type-specific. Another disadvantage of token-specific representations is that the corpus used to induce the HMM must have identical preprocessing as the data used during the supervised task.

Huang and Yates (2009) induce a fully-connected HMM, which emits a multinomial distribution over possible vocabulary words. The representation for a particular word token is the probability distribution over the states. Goldberg et al. (2009) use an HMM to assign POS tags to words, which in turns improves the accuracy of the PCFG-based Hebrew parser. Unlike (Huang & Yates, 2009), they use the emission probabilities, not the state probabilities, as the word representation. Li and McCallum (2005) use an HMM-LDA model to improve POS tagging and Chinese Word Segmentation.

At publication time, we were not able to include any HMM word representations in our experiments.

3 Distributed representations

Another approach to word representation is to learn a distributed representation. A distributed representation is dense, low-dimensional, and real-valued. Distributed word representations are called *word embeddings*. Each dimension of the embedding represents a latent feature of the word, hopefully capturing useful syntactic and semantic properties. A distributed representation is compact, in the sense that it can represent an exponential number of clusters in the number of dimensions.

Word embeddings are typically induced using *neural language models*, which use neural networks as the underlying predictive model (Bengio, 2008). By comparison, clustering-based word representations are usually induced under n-gram language models. Historically, training and testing of neural language models has been slow, scaling as the size of the vocabulary for each model computation (Bengio et al., 2001; Bengio et al., 2003). However, many approaches have been proposed in recent years to eliminate that linear dependency on vocabulary size (Morin & Bengio, 2005; Bengio & Sénécal, 2008; Collobert & Weston, 2008; Mnih & Hinton, 2009) and allow scaling to very large training corpora.

3.1 Collobert and Weston (2008) embeddings

Collobert and Weston (2008) presented a neural language model that was fast to train, because the gradient of the loss was computed stochastically over a small sample of possible outputs, in a spirit similar to Bengio and Sénécal (2003). This neural model of Collobert and Weston (2008) was refined and presented in greater depth in Bengio et al. (2009).

The model is discriminative and non-probabilistic. Given an n-gram, the model concatenates the learned embeddings of the n words, and predicts a score for that n-gram by passing the concatenation through a single hidden layer net. The training criterion is that n-grams that are present in the training corpus must have a score at least some margin higher than corrupted n-grams. In Collobert and Weston (2008), the middle word in the n-gram is corrupted. In Bengio et al. (2009), the last word in the n-gram is corrupted.

We implemented the approach of Collobert and Weston (2008), with the following differences:
• We did not achieve as low log-ranks on the English Wikipedia as the authors reported in Bengio et al. (2009), despite initially attempting to have identical experimental conditions.
• We corrupt the last word of the n-gram during training.
• We had a separate learning rate for the embeddings and for the rest of the model weights. We found that the embeddings should have a learning rate generally 1000–32000 times higher than the remaining model weights. Otherwise, the unsupervised training criterion drops slowly.
• Although their sampling technique makes training fast, testing is still expensive when the size of the vocabulary is large. Instead of cross-validating using the log-rank over the validation data as they do, we instead used the moving average of the training loss on training examples before the weight update.

3.2 HLBL embeddings

The log-bilinear model (Mnih & Hinton, 2007) is a probabilistic and linear neural model. Given an n-gram, the model concatenates the embeddings of the $n-1$ first words, and learns a linear model to predict the embedding of the last word. The similarity between the predicted embedding and the current actual embedding is transformed into a probability by exponentiating and then normalizing. Mnih and Hinton (2009) speed up model evaluation during training and testing by using a hierarchy to exponentially filter down the number of computations that are performed. This hierarchical evaluation technique was first proposed by Morin and Bengio (2005). The model, combined with this optimization, is called the *hierarchical log-bilinear (HLBL)* model.

4 Approach to named entity recognition

NER is typically treated as a sequence prediction problem. Following Ratinov and Roth (2009), we use the regularized averaged perceptron model. Ratinov and Roth (2009) showed that the BILOU encoding outperforms BIO, and the greedy inference performs competitively to Viterbi while being

significantly faster. Accordingly, we used greedy inference and BILOU text chunk representation. We used the publicly available implementation from Ratinov and Roth (2009), but we did not use gazetteers or non-local features.

After each epoch over the training set, we measured the accuracy of the model on the development set. Training was stopped after the accuracy on the development set did not improve for 10 epochs, generally about 50–80 epochs total. The epoch that performed best on the development set was chosen as the final model.

We use the following baseline set of features from Zhang and Johnson (2003):
• Previous two predictions y_{i-1} and y_{i-2}
• Current word x_i
x_i word type information: all-capitalized, is-capitalized, all-digits, alphanumeric, etc.
• Prefixes and suffixes of x_i, if the word contains hyphens, then the tokens between the hyphens
• Tokens in the window $c = (x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2})$
• Capitalization pattern in the window c
• Conjunction of c and y_{i-1}.

When using the lexical features, we normalize dates and numbers. For example, *1980* becomes **DDDD** and *212-325-4751* becomes **DDD*-*DDD*-*DDDD**. This allows a degree of abstraction to years, phone numbers, etc. This delexicalization is performed separately from using the word representation. That is, if we have induced an embedding for *12/3/2008* , we will use the embedding of *12/3/2008* , and **DD*/*D*/*DDDD** in the baseline features listed above.

5 Data

5.1 Unlabeled Data

Unlabeled data is used for inducing the word representations. We used the RCV1 corpus, which contains one year of Reuters English newswire, from August 1996 to August 1997, about 63 millions words in 3.3 million sentences. We left case intact in the corpus. By comparison, Collobert and Weston (2008) lowercases words and delexicalizes numbers.

Liang, (2005, p. 51) proposes a preprocessing technique, which was later used by Koo et al. (2008): Remove all sentences that are less than 90% lowercase a–z. We assume that whitespace is not counted, although this is not specified in Liang's thesis. We call this preprocessing step *cleaning*, but we leave the RCV1 corpus unclean except where noted. After cleaning, there are 37 million words (58% of the original) in 1.3 million sentences (41% of the original).

The RCV1 corpus has 651K word types. The cleaned RCV1 corpus has 269K word types. This is the vocabulary size, i.e. how many word representations were induced.

There were 405 word tokens (124 word types) in the CoNLL03 labeled data that do not appear in the RCV1 vocabulary.[1] There were 3082 word tokens (1507 word types) in the CoNLL03 labeled data that do not appear in the cleaned RCV1 vocabulary.

5.2 Labeled Data

The standard evaluation benchmark for NER is the CoNLL03 shared task dataset drawn from the Reuters newswire. The training set contains 204K words (14K sentences, 946 documents), the test set contains 46K words (3.5K sentences, 231 documents), and the development set contains 51K words (3.3K sentences, 216 documents). Note that cleaning is applied only to the unlabeled data, not to the labeled data (training nor dev not test).

RCV1 is a superset of the CoNLL03 corpus. For this reason, results that use RCV1 embeddings are a form of transductive learning.

[1] Article titles are present in the CoNLL03 data but were stripped from RCV1 during preprocessing, so a few uppercase versions of words were in the CoNLL03 data but had no representation.

Representation	#feat	Factor
baseline	76	1.0
Brown clusters	130	1.7
C&W embeddings	749	9.9

Table 1: Mean number of features per NER example. Column 3 is the multiplicative factor over the baseline. C&W is Collobert and Weston (2008). The baseline does not use any word representations. HLBL embeddings are twice the size of C&W embeddings.

6 Experiments and Results

6.1 Details of inducing word representations

The Brown clusters took roughly 3 days to induce. We induced 1000 clusters, as did prior work (Koo et al., 2008; Ratinov & Roth, 2009). Because Brown clusters are hierarchical, we can use cluster supersets as features. We used clusters at path depth 4, 6, 10, and 20 (Ratinov & Roth, 2009).

The Collobert and Weston (2008) embeddings were induced over the course of a few weeks. One of the difficulties in inducing these embeddings is that there is no stopping criterion defined, and that the quality of the embeddings can keep improving as training continues. Collobert (p.c.) simply leaves one computer training his embeddings indefinitely. We induced embeddings with 50 dimensions over 5-gram windows. We found that learning rate 3.2e-6 for the embeddings and 1e-10 for the neural network reduced the training loss most quickly on the unclean RCV1 corpus.

The HLBL embeddings took 6 days to induce. Unlike our Collobert and Weston (2008) embeddings, we did not extensively tune the learning rates for HLBL. We used a learning rate of 1e-3 for both model parameters and embedding parameters. We induced embeddings with 100 dimensions over 5-gram windows. Embeddings were induced over one pass approach using a random tree, not two passes with an updated tree and embeddings re-estimation.

6.2 Training time

Table 1 compares the number of active features per NER example for the different word representations. All word representations took roughly 50–80 training epochs to converge, and training time for an epoch is proportional to the number of active features. Hence, training with Collobert and Weston (2008) embeddings took an order of magnitude more time than training the baseline. (The baseline trained on the order of a few hours.)

6.3 Normalization of Word Embeddings

Like many NLP systems, the baseline system contains only binary features. The word embeddings, however, are real numbers that are not necessarily in a bounded range. If the range of the word embeddings is too large, they will exert more influence than the binary features.

Assume that the embeddings are represented by a matrix E, where row i, E_i, is the representation of word i. We experimented with two strategies for normalizing the embeddings:
independent We normalize each embedding independently: $E_i \leftarrow E_i/(a \cdot \max(|E_i|))$
overall We normalize each embedding under the same factor: $E_i \leftarrow E_i/(a \cdot \max(|E|))$
a is a normalization constant. Infrequent words tend to have a lower norms, because they have been updated less frequently during training. Therefore, the independent normalization will scale rare words to be as important as the more common words, while overall normalization will have the model give more importance to the embeddings of more frequent words.

In preliminary experiments, we tried both normalization strategies with a variety of constants a. We induced embeddings on the English Wikipedia (1.5B words) using the approach of Collobert and Weston (2008). We converted the corpus to lowercase, and induced embeddings on the most frequent 20K words. The values of these embeddings were within the range $[-15, +12]$ with mean 0.01 and stddev 1.15. (In all other experiments in this paper, we used the RCV1 corpus, case-intact except where noted, and did not delexicalize any words.) We also used the HLBL embeddings induced on

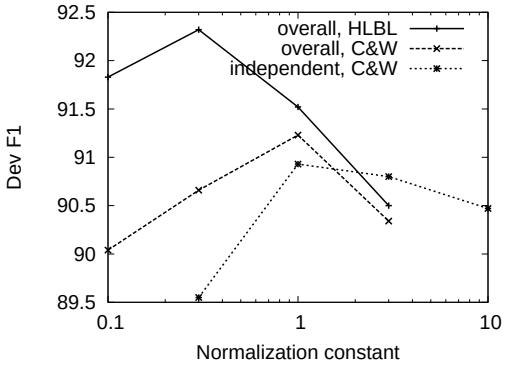

Figure 1: Effect of different normalization strategies and parameters on NER dev F1. HLBL embeddings were induced on (unclean) RCV1. Collobert and Weston (2008) embeddings were induced on the English Wikipedia with a 20K vocabulary, for this experiment only. (All other experiments used embeddings induced over RCV1.)

Representation	Clean	Dev F1	Test F1
Brown clusters	✓	92.47	88.39
Brown clusters		91.65	88.10
HLBL embeddings		92.32	87.66
C&W embeddings	✓	91.77	87.13
C&W embeddings		91.14	86.27
none (baseline)	n/a	89.87	84.07

Table 2: NER F1 on the dev set and test set, using different representation trained on RCV1. Some word representations were induced over the cleaned RCV1, as indicated by the second column. C&W is Collobert and Weston (2008). As of publication time, we were not able to train HLBL on the cleaned RCV1.

RCV1 as described in Section 5.1, which were within the range $[-1.21, +0.92]$ with mean 0.00 and stddev 0.05.

Figure 1 shows the effect of normalization technique on the NER system. This table indicates that the accuracy of the NER system is sensitive to the choice of this value, and should be optimized. Overall normalization with constant $a = 1.0$ had the best dev set performance with Collobert and Weston (2008) embeddings, and this is what we used for inducing Collobert and Weston (2008) embeddings for the rest of the experiments.[2] For HLBL embeddings, we used normalization constant $a = 0.3$. Neither of these normalization constants corresponds to transforming the embeddings to have unit variance, so we cannot prescribe any analytical technique for choosing the normalization constant.

6.4 Final results

Table 2 shows the final NER F1 results on the test set, using different representation trained on RCV1.

Brown clusters provide a larger accuracy increase over the baseline than both styles of embeddings. HLBL embeddings outperform Collobert and Weston (2008) embeddings, despite taking an order of magnitude less time to induce. HLBL embeddings are induced under a linear model, whereas Collobert and Weston (2008) embeddings are induced under a non-linear classifier. We hypothesize that, for this reason, HLBL embeddings are more suitable as features when the classifier for the supervised task is linear, like the perceptron we use in NER. We do not know why the embeddings experience a steeper drop from dev F1 to test F1 than the Brown clusters. It is not clear what the results would be if we included additional features used in NER, such as POS tags, shallow parsing information, and gazetteers.

The clean corpus gave better representations, both with Collobert and Weston (2008) embeddings and Brown clusters. This is the case even though the cleaning process was very aggressive, and

[2]It is possible that a different normalization constant is needed when we change the unlabeled corpus to RCV1.

discarded more than half of the sentences. According to the evidence and arguments presented in Bengio et al. (2009), the non-convex optimization process for Collobert and Weston (2008) embeddings may be adversely affected by noise and the statistical sparsity issues regarding rare words, especially at the beginning of training. For this reason, we hypothesize that learning representations over the most frequent words first and gradually increasing the vocabulary (a *curriculum* training strategy), would provide better results than cleaning (Bengio et al., 2009).

When there is not much contextual evidence for the current label decision, the model can "back-off" to its estimate Pr(label|token). We hypothesize that this probability is more difficult to estimate under a linear model (Perceptron) when the representation for each token is low-dimensional and dense (embeddings). High-dimensional and sparse representations appear preferable when many of the decisions we make rely upon words that are the least common (named entities) in the vocabulary. It might be the case that distributed representations are more favorable in situations like parsing, where syntactic decisions are usually based upon common words (function words like "of", "at", etc.) and medium rareness words (nouns, verbs) etc. It might also be the case that Brown clustering is not a good method for inducing word representations for function words. However, error analysis does indicate that Brown clustering is better than distributed representations for modeling fine-grained semantics using a linear model.

7 Conclusions

Word embeddings seem more promising than Brown clusters, because Brown clusters only model bigram statistics whereas word embeddings model relationships over an entire n-gram window. However, in our experiments Brown clusters outperform both styles of word embeddings.

The distributed representation of embeddings is compact, in the sense that it can represent an exponential number of clusters in the number of dimensions. Perhaps this compactness is not as good for features of a linear supervised model, and a high-dimensional sparse representation is better? It might also be the case that if our supervised model used a non-linear classifier (SVM, neural network, etc.), that the embeddings' distributed representations are more appropriate than class-based representations.

We were surprised to find HLBL embeddings outperform Collobert and Weston (2008) embeddings, despite taking less time to induce.[3] HLBL embeddings are induced under a linear model, whereas Collobert and Weston (2008) embeddings are induced under a non-linear model. We hypothesize that, for this reason, HLBL embeddings are more suitable as features when the classifier for the supervised task is linear, like the perceptron we use in NER. We did not experiment with different learning rates for inducing the HLBL embeddings. We are curious if HLBL embeddings would be competitive with Brown clusters, if we were to tune these learning rates.

There are other difficulties in working with word embeddings:

- There is no clearly defined stopping criterion. One can train the embeddings indefinitely, and we have trained the Collobert and Weston (2008) embeddings for weeks. Brown clustering, by comparison, stops when each word has been inserted into a cluster.
- Embeddings, because they are dense representations, increase the number of active features in the model by far more than Brown clusters. This, in turn, increases supervised training time.
- Brown clusters have no hyperparameters that we needed to tune. (We chose 1000 clusters, the standard choice in the literature.)
- When inducing Collobert and Weston (2008) embeddings, we must choose both a learning rate for the embeddings and a learning rate for the rest of the parameters. This might also be helpful when inducing HLBL embeddings.
- Embeddings are sensitive to choice of normalization, which introduces another hyperparameter.

The clean corpus gave better representations, both with Collobert and Weston (2008) embeddings and Brown clusters. We are interested to understand better how to induce representations, so that they are less sensitive to noise in the unlabeled corpus.

[3] Preliminary experiments showed that HLBL embeddings after 1.5 days of training are already better than the Collobert and Weston (2008) embeddings, despite being induced for an order of magnitude less time.

We are curious how these representations will work as word features on other NLP tasks. This work is part of a larger survey work that is underway: We are currently collaborating with other authors who have near-state-of-the-art baseline systems, and adding our word representations to them. We encourage researchers who are interested in improving the accuracy of their systems to contact us about a potential collaboration.

Acknowledgments

Thank you to Alexander Yates for useful discussion. Thank you to Andriy Mnih for inducing his embeddings on RCV1 for us.

References

Bengio, Y. (2008). Neural net language models. *Scholarpedia*, *3*, 3881.

Bengio, Y., Ducharme, R., & Vincent, P. (2001). A neural probabilistic language model. *NIPS*.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*, 1137–1155.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. *ICML*.

Bengio, Y., & Sénécal, J.-S. (2003). Quick training of probabilistic neural nets by importance sampling. *AISTATS*.

Bengio, Y., & Sénécal, J.-S. (2008). Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, *19*, 713–722.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., & Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, *18*, 467–479.

Candito, M., & Crabbé, B. (2009). Improving generative statistical parsing with semi-supervised word clustering. *IWPT* (pp. 138–141).

Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. *ICML*.

Goldberg, Y., Tsarfaty, R., Adler, M., & Elhadad, M. (2009). Enhancing unlexicalized parsing performance using a wide coverage lexicon, fuzzy tag-set mapping, and EM-HMM-based lexical probabilities. *EACL*.

Huang, F., & Yates, A. (2009). Distributional representations for handling sparsity in supervised sequence labeling. *ACL*.

Koo, T., Carreras, X., & Collins, M. (2008). Simple semi-supervised dependency parsing. *ACL* (pp. 595–603).

Li, W., & McCallum, A. (2005). Semi-supervised sequence modeling with syntactic topic models. *AAAI*.

Liang, P. (2005). Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology.

Lin, D., & Wu, X. (2009). Phrase clustering for discriminative learning. *ACL* (pp. 1030–1038).

Martin, S., Liermann, J., & Ney, H. (1998). Algorithms for bigram and trigram word clustering. *Speech Communication*, *24*, 19–37.

Miller, S., Guinness, J., & Zamanian, A. (2004). Name tagging with word clusters and discriminative training. *HLT-NAACL* (pp. 337–342).

Mnih, A., & Hinton, G. E. (2007). Three new graphical models for statistical language modelling. *ICML*.

Mnih, A., & Hinton, G. E. (2009). A scalable hierarchical distributed language model. *NIPS* (pp. 1081–1088).

Morin, F., & Bengio, Y. (2005). Hierarchical probabilistic neural network language model. *AISTATS*.

Ratinov, L., & Roth, D. (2009). Design challenges and misconceptions in named entity recognition. *CoNLL*.

Schwenk, H., & Gauvain, J.-L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 765–768). Orlando, Florida.

Ushioda, A. (1996). Hierarchical clustering of words. *COLING* (pp. 1159–1162).

Zhang, T., & Johnson, D. (2003). A robust risk minimization based named entity recognition system. *CoNLL*.

Zhao, H., Chen, W., Kit, C., & Zhou, G. (2009). Multilingual dependency learning: a huge feature engineering method to semantic dependency parsing. *CoNLL* (pp. 55–60).