Context Representation with Word Embeddings for WSD

Hiromu Sugawara[1], Hiroya Takamura[2], Ryohei Sasano[2(✉)], and Manabu Okumura[2]

[1] Department of Information Processing, Tokyo Institute of Technology, Tokyo, Japan
suga@lr.pi.titech.ac.jp
[2] Precision and Intelligence Laboratory, Tokyo Institute of Technology, Tokyo, Japan
{takamura,sasano,oku}@pi.titech.ac.jp

Abstract. Word embeddings obtained through neural language models developed recently can capture semantic and grammatical behaviors of words and very capably find relationships between words. Such word embeddings are shown to be effective for various NLP tasks. In this paper, we develop a supervised method for word sense disambiguation (WSD) that employs word embeddings as local context features. Our experiments show the usefulness of word embeddings in the WSD task. We also compare the methods with different vector representations and reveal their effects on the WSD task.

Keywords: Word sense disambiguation · Word representation · Supervised machine learning

1 Introduction

Polysemous words are a major obstacle in many natural language processing (NLP) tasks. To circumvent this obstacle, NLP researchers have been developing methods for word sense disambiguation (WSD) [3,8,12,22]. Supervised learning approaches have performed especially well in many NLP tasks including the WSD task. Since the words in the neighborhood/context of the target polysemous word provide clues to the disambiguation, bag-of-words (BoW) of the context (usually a few words preceding or following the target word) is often used as a basic feature set. In addition to the local context features such as the BoW, there are several features that have been usually used in supervised WSD: topical features, syntactic features, and semantic features [18]. Other features also exist for the WSD task. For example, Agirre et al. [1] used WordNet [17] as a resource for representing domains in WSD and constructed two domain features: the domain that is the most relevant with the context, and a list of domains the relevance of which is above a predefined threshold. It has been expected that combining these additional features with the local context features will contribute to the improvement of the system performance.

© Springer Science+Business Media Singapore 2016
K. Hasida and A. Purwarianti (Eds.): PACLING 2015, CCIS 593, pp. 108–119, 2016.
DOI: 10.1007/978-981-10-0515-2_8

When we use BoW as local context features, the context words in the test data that do not appear in the training data cannot be effective, even if their synonymous words are found in the training data. One of the approaches which attacks the problem of the sparseness is using concepts of context words rather than the words themselves as features. In this approach, the concepts are usually based on a thesaurus or a knowledge base. The other way is to use vector representation of words as representation of local context features. When we use the vector representation, cosine similarity between two vectors is usually used for calculating similarity between two words. Therefore, we can say that two words are similar when their two vector representations look similar. To solve the above-mentioned problem, in this paper, we attempt to represent the sense of each context word by means of the vector representations of words.

Many researchers have attempted to represent the word meanings as real-valued vectors. Recently, neural language models, such as the feed-forward neural network language model [5] and the recurrent neural network language model [15], have succeeded in obtaining the word representation (or embedding) that captures the semantic and grammatical behaviors of words. There are a number of implementations and variants of those neural language models above. Among them, Mikolov et al. proposed a skip-gram model and a continuous bag-of-words (CBOW) model [14,16]. Both the skip-gram model and the CBOW model are log-linear models without any nonlinear hidden layer. The word embeddings obtained by the skip-gram model and the CBOW model have been shown to be very useful to calculate word similarity. In particular, the word embeddings obtained by the skip-gram model achieved the best performance in their experiments.

The high performance in measuring word similarity suggests that the word embeddings obtained through the neural language models are good candidates for the vector representations of context words for WSD. Therefore, we employ word embeddings obtained by neural language models as features for supervised sense classifiers and confirm the usefulness of word embeddings in a WSD task. To support this claim, we also compare the word embeddings by neural language models with other vector representations in terms of the accuracy of WSD.

2 Background

The skip-gram model is a language model proposed by Mikolov et al. [14,16]. This language model predicts words that appear within a context window of an input word, which consists of N_e word tokens to the left and another N_e tokens to the right.

The skip-gram model differs from the models based on simple co-occurrences of words in the context window, because it assumes that each word in the context window, as well as the word in the center ("walking" in the example below), also has its own embedding and that the co-occurrence is caused by the two embeddings. Therefore, this model learns vector representations of words to assign larger co-occurrence probability to word pairs co-occurring more frequently.

The skip-gram model has two different types of parameters: *input* and *output* vector representations for each word. The first type of vector represents the word w_I in the center of the context window, while the second type of vector represents the other words w_O in the context window. The probability of w_O appearing within a context-window of w_I is defined as follows:

$$p(w_O|w_I) = \frac{\exp(\boldsymbol{v}'_{w_O} \cdot \boldsymbol{v}_{w_I})}{\sum_{w \in W} \exp(\boldsymbol{v}'_w \cdot \boldsymbol{v}_{w_I})} \tag{1}$$

where $\boldsymbol{v}_w$ and $\boldsymbol{v}'_w$ are the vector representations of *input* and *output* of w, and W is the vocabulary. Thus, this model assigns a large conditional probability to word pair w_I and w_O if the inner product of $\boldsymbol{v}_{w_I}$ and $\boldsymbol{v}'_{w_O}$ is large.

Specifically, the model maximizes the average of the logarithm of probability:

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-N_e \leq j \leq N_e, j \neq 0} \log p(w_{t+j}|w_t) \tag{2}$$

where T is the number of words in the training corpus and w_t is the t-th word in the corpus.

For example, suppose that there are two sentences in the training corpus:

- The cat is walking in the bedroom.
- The cat was running in the room.

If N_e is set to 2, the words in the context window of "walking" are "cat", ""is", "in", and "the" in this example. Also, the words in the context window of "running" are "cat", "was", "in", and "the". The skip-gram model increases the probabilities of each pair of the word in the center (e.g.,"walking" and "running") and a context word (e.g., "cat" for both examples) by bringing the *input* vector of the word in the center and the *output* vector of the context words close together, i.e. the cosine similarity between these vectors becomes large. Thus, the vector pair ($\boldsymbol{v}_{walking}$ and $\boldsymbol{v}'_{cat}$) get close to each other. The pair ($\boldsymbol{v}_{running}$ and $\boldsymbol{v}'_{cat}$) also get close to each other. As a result, *input* vectors of words with similar context ($\boldsymbol{v}_{walking}$ and $\boldsymbol{v}_{running}$) become similar to each other through the *output* vector ($\boldsymbol{v}'_{cat}$).

Word embeddings are usually used as the input layer of neural network models [10,21]. Collobert and Weston [10] proposed a single convolutional neural network architecture that performs multi-task learning. Their model learns each word embedding as feature representations of each task. Although these (deep) neural network models work very well, the computational cost is mostly high. It is therefore practically important to consider how to represent instances of the data in supervised learning by using word embeddings learned from an unlabeled corpus, because feature representations are fundamental to supervised classifiers that have lower computational cost. Yu et al. [23] employed the feature that represents words by a cluster of embeddings. They used the support vector machines (SVM) and the multilayer perceptron with their features on chunking and named entity recognition tasks.

Some researchers have extended vector space models to deal with polysemous words. Agirre et al. [2] proposed a method of using the inter-word similarity

on the basis of latent semantic analysis (LSA). They applied the method to the domain adaptation of word sense disambiguation. Cai et al. [7] used latent Dirichlet allocation (LDA) [6] to create a naive Bayesian classifier, and achieved a high accuracy in the Semeval 2007 coarse-grained lexical sample task [20]. Chen et al. [9] proposed a unified model for joint word sense representation and disambiguation, which assigned a distinct representation for each word sense. Their model achieved state-of-the-art performance on the coarse-grained all-words dataset and domain-specific WSD dataset. Neelakantan et al. [19] also focused on a method for obtaining word sense representation. They proposed two models: one has a fixed number of word senses, and the other automatically determines the number of word senses.

3 Methodology

In this section, we first describe the task definition of WSD and then present the features of the supervised classifier that employ word embeddings.

3.1 Task Definition

The word sense disambiguation is a task to choose appropriate senses of polysemous words in the given context. The possible senses of each word are based on external knowledge, such as WordNet, in most cases. In the lexical sample task of WSD, the corpus annotated with word senses is usually given as the training data, and the WSD task is reduced to a supervised classification task. Since a single polysemous word can have three or more senses, we should construct a multi-class classifier.

3.2 Proposed Feature Representation

For a classifier, we use the support vector machines (SVM) together with the one-versus-rest approach to extend SVMs to a multi-class classifier. In this section, we explain feature representation based on word embeddings and we call it Context Word Embeddings. Since we compare four types of context features in our experiments in the next section, we explain them all in this section. Two are based on BoW, and the others are based on word embeddings.

Bag-of-Words (BoW)
: This type of feature indicates whether a word appears within the context window of size N. The dimension of the feature space is equal to the vocabulary size $|W|$.

Position-Bag-of-Words (PosiBoW)
: With this feature set, each feature vector is represented as a concatenation of one-of-V representations. One-of-V representation represents each word by a binary vector, in which only the element associated with this word is 1, and the others are 0. The one-of-V representations of words within the

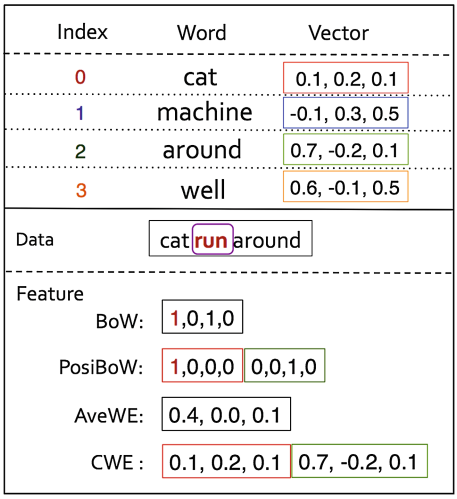

Fig. 1. Example of each feature

context window are concatenated to make a feature vector of $2 \times N \times |W|$. While the simple BoW described above does not contain position information, PosiBoW does. We use this feature for comparison, because the following Context-Word-Embeddings feature takes positions into consideration.

Average-Word-Embeddings (AveWE)
This feature set uses word embeddings, but does not take into account the position of each context word. The feature vector is the average of vector representations of words in the context window. The dimension of the feature space is the same as the dimension of each word embedding.

Context-Word-Embeddings (CWE)
This feature vector is a concatenated vector of the real-valued vectors of the words in the context window. If the window size is N and words appearing in the context window are $w_{-N}, \ldots, w_{-1}, w_{+1}, \ldots, w_{+N}$, this feature vector is a vector concatenating $v_{w_{-N}}, \ldots, v_{w_{-1}}, v_{w_{+1}}, \ldots, v_{w_{+N}}$, where v_w represents an embedding of word w. If the dimension of each word embedding is d, the size of this feature vector is $2 \times N \times d$.

Fig. 1 shows a simple example of each feature representation. Please assume that only four words "cat," "machine," "around," and "well" are in the vocabulary (the vocabulary size $|W|$ is 4). In the top, word embeddings for each word are shown (the dimension of each word embedding d is 3). The instance data to be represented as feature representation is shown in the middle, where the target word is "run" and the window size N is 1. In the bottom, four types of context features are shown.

4 Experiments

We evaluated word embedding based features on an English lexical-sample data set. We also investigated the effect of the difference in word embeddings.

4.1 Experimental Settings

We used the SemEval 2007 lexical sample task (task17) dataset [20]. This dataset contains training and test data for 100 polysemous words. The average numbers of instances are 222 for the training set and 48 for the test set.

We lemmatized each word by using the lemmatizer in the Natural Language Toolkit[1] that is based on WordNet. We chose the skip-gram model [16] to learn embeddings. Mikolov et al. [16] distributed word embeddings learned from news articles containing about 100 billion words[2]. We used this data to compare features based on binary vector and features based on word embeddings. The dimension of these vectors is 300. We also used linear SVM (LIBLINEAR [11]) as a classifier and used five-fold cross validation on the training dataset in order to determine the value of soft-margin parameter C by changing its value from 0.1 to 1 with a step size of 0.1 and from 1 to 10 with a step size of 1.

4.2 Comparison of BoW and CWE

We used four types of features explained in Sect. 3: BoW, PosiBoW, AveWE, and CWE. We also tested their combinations in experiments. Table 1 shows the experimental results.

Table 1. Classification results of each feature and combination of features

Features	Accuracy
BoW	84.72 %
PosiBoW	85.53 %
AveWE	84.56 %
CWE	87.51 % † ‡
PosiBow+CWE	87.18 % † ‡
BoW+CWE	87.80 % † ‡

'†' and '‡' denote significant differences from BoW and PosiBoW, respectively.

We performed McNemar's test [13] at the significance level of 1 % to assess whether two classifiers were performing significantly differently. '†' and '‡' mean

[1] http://www.nltk.org/.
[2] https://code.google.com/p/word2vec/.

that the corresponding feature set significantly outperforms BoW and PosiBoW, respectively. This result shows that CWE is better than BoW, PosiBoW, and AveWE in this task. This also shows that PosiBoW outperforms BoW. Therefore, position information of words would be helpful on WSD. However, the system using PosiBoW+CWE performed worse than that using BoW+CWE. We conjecture that PosiBoW+CWE does not work better than BoW+CWE despite the useful information from word positions, because CWE itself contains position information. CWE worked very well, but AveWE did not work well. There are two possible reasons: one is that the size of context window was not enough to represent context, and the other is that simply averaging vectors can not represent context well. Actually, we tried AveWE with broader context windows, but obtained worse results. Thus we consider this is because averaging word-representations obscures information about which words appear in local context. We conclude simple averaging is not enough to represent context from this result. Actually, we tried AveWE with broader context window, but it shows worser result. Thus, we consider this is because averaging word-representations lose information about what words appear in local context. We conclude simple averaging is not enough to represent context from this result.

If many words in an instance data to be classified do not appear in the training dataset, it is difficult to classify the instance correctly, especially when we use BoW. However, when we use word embeddings, information of words that do not appear in the training set is expected to be leveraged for classification. To confirm this assumption, we checked the number of words that do not appear in the context window when we train classifiers for each instance in the test set. Hereinafter, we call such words unknown word (UNKs), and we also examine the correlation between the accuracy and the number of UNKs.

Table 2 shows the number of instances per the number of UNKs and Fig. 2 shows the ratios (%) of correct and incorrect outputs per the number of UNKs. The first figure shows the ratios with BoW only, and the second figure shows the ratios of the system using BoW and CWE.

Table 2. The number of instances per the number of UNKs

0	1	2	3	4	5	6	7	8	9	10
564	765	872	846	766	525	321	109	50	21	12

The first and second rows denote the number of UNKs and the number of instances, respectively.

Vertical and horizontal axes of both graphs are the ratios and the number of UNKs, respectively. Thus, the rightmost bar represents cases in which all words are UNKs, and the leftmost bar represents cases in which neither words are UNKs. Right cases are expected to be more difficult to classify than left cases. When we used only BoW, the ratio of incorrect outputs increased as we had assumed. Next, we focus on bars representing the ratios of cases whose result

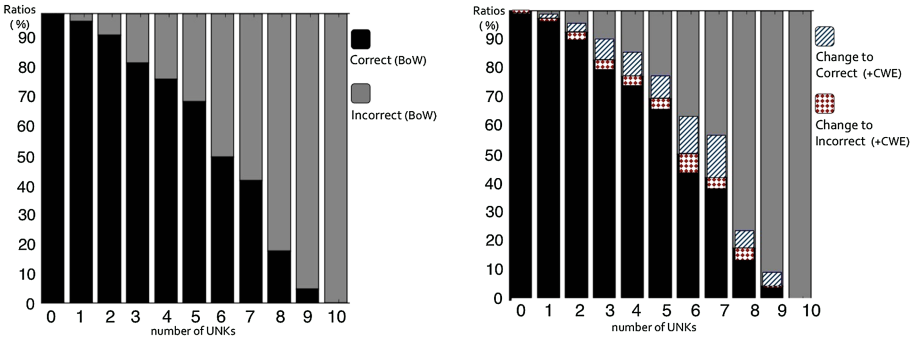

Fig. 2. Effect of UNKs on accuracy

Table 3. The relationships between the accuracy and the sizes of training sets

Feature	The size of training set			
	100 %	75 %	50 %	25 %
PosiBoW	85.53 %	83.67 %	82.81 %	80.64 %
CWE	87.51 %	86.94 %	86.44 %	84.85 %
diff.	1.98 %	3.27 %	3.63 %	4.21 %

became correct when we used BoW and CWE (bars with blue slash). These ratios did not monotonically decrease even if the number of UNKs increased. These results show that CWE can utilize of UNKs, while BoW cannot.

The feature set based on word embeddings would alleviate the sparseness problem. Thus, we next examined how the performance of the classifier changed depending on the size of training data. The size of test data was the same as in the above experiment, although we changed the size of training data in this experiment. We randomly divided each piece of training data of each word into quarters and then gradually removed quarters from training data. Each piece of divided data has almost the same amount of data for each sense. Table 3 shows the relationships between the accuracy and the number of the training instances. The row *diff.* shows the accuracy differences between PosiBoW and CWE. Table 3 shows that the difference between two systems is large when the number of training instances is small. This result suggests that the CWE feature is useful, especially when only a small training data set is available.

4.3 Effect of the Methods for Constructing Vector Representations

A method for constructing vector representation is crucial for a high performance of WSD. To investigate how the choice of vector representation affects the performance of WSD, we used the singular value decomposition (SVD) and word2vec[3] (skip-gram model) to obtain vector representation. We also used

[3] https://code.google.com/p/word2vec/.

Table 4. Classification results of each word representation

	CWE	BoW + CWE
EnWiki SVD	83.54 %	86.15 %
EnWiki w2v	86.45 %	86.58 %
Google w2v	87.51 %	87.80 %
(BoW)	84.72 %	

English Wikipedia[4] data to obtain vector representations of words. This data contains 1.7 billion words. This is 0.5 % size of the corpus used in learning vectors distributed by Mikolov et al. [16].

We chose a method using SVD following Baroni et al. [4]. We regarded two word tokens as having co-occurred when their distance was less than N_e. We then made the co-occurrence matrix by using their method and factorized the matrix as $X = U\Sigma V^{\mathrm{T}}$. In this study, X represents the co-occurrence matrix. We assume that each row of U represents the vector of the word associated with the row. When we used word2vec, we chose the skip-gram with negative-sampling as a model and set the number of negative-samples to 10. We also set the dimension of vectors to 300 both for skip-gram and SVD.

We compared three vector representations: vectors obtained by SVD from English Wikipedia (EnWiki SVD), vectors obtained by word2vec from English Wikipedia (EnWiki w2v), and vectors distributed by Mikolov et al. that are trained on part of the Google News dataset (Google w2v). Table 4 shows the experimental results.

With the word representation by SVD, the system using both BoW and CWE (BoW+CWE) outperformed BoW, although CWE did not outperform BoW. On the other hand, with the word representation by word2vec, both CWE and BoW+CWE outperformed BoW even when using English Wikipedia data to obtain the vector representation. This result suggests that the word embeddings obtained by the skip-gram model contain very helpful information for WSD.

4.4 Examples of Word Embeddings Affecting the Results

Table 5 shows instances, for which BoW predicted the wrong sense and BoW+CWE with Google w2v predicted the correct sense. The blue italicized words in the table represent UNKs, and the red bold words are the target words. In the examples of the sense 2 of the noun "management", which means *the people who direct a business*, there are some UNKs that are names of companies or organizations: "Younkers", "swift", and "Wedtech". Well-learned embeddings would capture the similarity between these names appearing in training data and test data, resulting in a high performance of CWE for "management".

UNKs could also be clues for predicting the sense in the examples of the verb "begin". Senses shown in the examples of "begin" are all sense 2, i.e., *take*

[4] We accessed the Wikipedia dataset in August 2014.

Table 5. Examples improved by using CWE

Word	Sense	Examples improved by using CWE
management.n	2	*Younkers* **management** is likely to buy a 10 % to 20 % interest in the chain in January , said Fred S. Hubbell , Equitable 's president and chief executive officer .
	2	Subcontractors will be offered a settlement and a *swift transition* to new **management** is *expected* to *avert* an exodus of skilled workers from Waertsilae Marine 's two big shipyards , government officials said .
	2	*Wedtech* **management** *used* the *merit* system .
	2	New **management** at *Kentucky Fried Chicken*, a unit of PepsiCo Inc. , has fought back with new medium and large chicken sandwiches for the lunch crowd .
begin.v	2	If the investor does n't put up the extra cash to satisfy the call , the *brokerage* firm *may* **begin** *liquidating* the *securities* .
	2	General Motors Corp. said it had discussed the possibility of a joint *venture* with *Jaguar* before *Ford* **began** *buying* shares .
	2	Precision Castparts Corp. , *Portland* , *Ore.* , will **begin** trading with the *symbol PCP* .

the first step or steps in carrying out an action. In these examples, two gerunds ("buying" and "liquidating") are UNKs. However, since "trading" appears in the training data and the cosine similarities between the embeddings of "trading" and those gerunds are large, these examples were correctly disambiguated by means of CWE. Note that although these words are not synonyms, they are given a high similarity by CWE because they are semantically related and have the same type of inflection (i.e., -ing). We thus consider the system outputs of these instances changed to correct answers because CWE captured the similarity of these words.

5 Conclusion

In this paper, we investigated the effects of features based on word embeddings on a WSD task. We confirmed that the classifier based on the word embeddings feature set outperforms those based on bag-of-words features. Our experiments also confirmed that the feature set based on word embeddings was more robust to the sparseness problem than features based on binary representation. We expect that the features that we used will contribute to the performance of a supervised classifier by being combined with other features. Although the feature sets that take word positions into account (CWE and PosiBoW) outperformed those that do not, they also have a downside: they are sensitive to the slight difference in positions of context words. A better way to handle the positions of context words would improve the performance of WSD.

References

1. Agirre, E., de Lacalle, O.L.: UBC-ALM: Combining k-NN with SVD for WSD. In: Proceedings of the 4th International Workshop on Semantic Evaluations (Semeval) pp. 342–345 (2007)
2. Agirre, E., de Lacalle, O.L.: Supervised domain adaption for WSD. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL), pp. 42–50 (2009)
3. Banerjee, S., Pedersen, T.: An adapted lesk algorithm for word sense disambiguation using wordnet. In: Gelbukh, A. (ed.) CICLing 2002. LNCS, vol. 2276, pp. 136–145. Springer, Heidelberg (2002)
4. Baroni, M., Dinu, G., Kruszewski, G.: Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL), pp. 238–247 (2014)
5. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**, 1137–1155 (2003)
6. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. The Journal of Machine Learning Research **3**, 993–1022 (2003)
7. Cai, J.F., Lee, W.S., Teh, Y.W.: NUS-ML: improving word sense disambiguation using topic features. In: Proceedings of the 4th International Workshop on Semantic Evaluations. Association for Computational Linguistics (SemEval-2007), pp. 524–531 (2007)
8. Carpuat, M., Wu, D.: Improving statistical machine translation using word sense disambiguation. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 61–72 (2007)
9. Chen, X., Liu, Z., Sun, M.: A unified model for word sense representation and disambiguation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1025–1035 (2014)
10. Collobert, R., Weston, J.: A unified architecture for natural language processing: deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning (ICML), pp. 160–167 (2008)
11. Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J., Fan, R.E.: LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. **9**, 1871–1874 (2008)
12. Ide, N., Vronis, J.: Word sense disambiguation: The state of the art. Computational Linguistics **24**, 1–40 (1998)
13. McNemar, Q.: Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika **12**(2), 153–157 (1947)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Workshop at International Conference on Learning Representations (ICLR) (2013)
15. Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., Khudanpur, S.: Recurrent neural network based language model. In: 11th International Conference of the International Speech Communication Association (INTERSPEECH), pp. 1045–1048 (2010)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems 26 (NIPS), pp. 3111–3119 (2013)

17. Miller, G.A.: Wordnet: a lexical database for english. Commun. ACM **38**(11), 39–41 (1995)
18. Navigli, R.: Word sense disambiguation: a survey. ACM Comput. Surv. **41**(2), 10:1–10:69 (2009)
19. Neelakantan, A., Shankar, J., Passos, A., McCallum, A.: Efficient non-parametric estimation of multiple embeddings per word in vector space. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1059–1069 (2014)
20. Pradhan, S.S., Loper, E., Dligach, D., Palmer, M.: SemEval-2007 task 17: English lexical sample, SRL and all words. In: Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007), pp. 87–92 (2007)
21. Socher, R., Huang, E.H., Pennington, J., Ng, A.Y., Manning, C.D.: Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: Proceedings of Advances in Neural Information Processing Systems 24 (NIPS), pp. 801–809 (2011)
22. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics (ACL), pp. 189–196 (1995)
23. Yu, M., Zhao, T., Dong, D., Tian, H., Yu, D.: Compound embedding features for semi-supervised learning. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pp. 563–568 (2013)