Learning Word Sense Embeddings from Word Sense Definitions

Qi Li       Tianshi Li       Baobao Chang
Key Laboratory of Computational Linguistics, School of Electronics Engineering and Computer Science, Peking University Beijing, P.R.China, 100871
qi.li@pku.edu.cn lts_417@hotmail.com chbb@pku.edu.cn

Abstract

Word embeddings play a significant role in many modern NLP systems. Since learning one representation per word is problematic for polysemous words and homonymous words, researchers propose to use one embedding per word sense. Their approaches mainly train word sense embeddings on a corpus. In this paper, we propose to use word sense definitions to learn one embedding per word sense. Experimental results on word similarity tasks and word sense disambiguation task show that word sense embeddings produced by our approach are of high quality.

1 Introduction

With the development of the Internet and computation efficiency of processors, gigantic unannotated corpora can be obtained and utilized for natural language processing (NLP) tasks. Those corpora can be used to train distributed word representations (i.e. word embeddings) which play an important role in most state-of-the-art NLP neural network models. The word embeddings capture syntactic and semantic properties which can be exposed directly in tasks such as analogical reasoning (Mikolov et al., 2013b), word similarity (Huang et al., 2012) etc. Prevalent word embedding learning models include Skip-gram (Mikolov et al., 2013b), Glove (Pennington et al., 2014) and variants of them.

Basic Skip-gram (Mikolov et al., 2013b) and Glove (Pennington et al., 2014) output one vector for each word. However, multi-sense words (including polysemous words and homonymous words) should inherently have different embeddings for different senses. Therefore researchers propose to use one embedding per word sense (Reisinger and Mooney, 2010; Huang et al., 2012; Chen et al., 2014; Neelakantan et al., 2014; Tian et al., 2014; Iacobacci et al., 2015; Liu et al., 2015; Wu and Giles, 2015; Li and Jurafsky, 2015). Previous work tends to perform word sense induction (WSI) or word sense disambiguation (WSD) on the corpus to determine the senses of words. Then they train the word sense embeddings on it using variants of Skip-gram (Mikolov et al., 2013b) or other approaches. The result of WSI or WSD on the corpus is not reliable and the errors from WSI or WSD will have bad effect on the training of word sense embeddings. Besides, these approaches normally produce bad embeddings for rare word senses.

Lexical ontologies such as WordNet (Miller, 1992) and BabelNet (Navigli and Ponzetto, 2012) are built by specialists in linguistics and they provide semantic information of word senses including their definitions. Different from determining word senses by WSI or WSD models, semantic information provided by lexical ontologies is normally accurate and reliable. To utilize the accurate information of word senses provided by lexical ontologies, we propose an approach based on recurrent neural networks (RNN) to learn word sense embeddings from word sense definitions. Our approach learns both word sense embeddings and a definition understanding model. Since the summation of definitions is much smaller in scale than a corpus for embedding training, our approach is less time-consuming comparing with corpus-based learning approaches. Experimental results show that the word sense embeddings are of high quality for both common and rare words and the definition understanding model can understand other natural language text besides word sense definitions.

Our contributions can be summarized as follows:

- We propose to learn word sense embeddings from word sense definitions using RNN-based models.
Different from previous embedding learning approaches, our learning is conducted in a supervised paradigm.

Our approach is less time-consuming comparing with corpus-based learning approaches.

Our approach trains rare word senses equally which is hard for corpus-based learning approaches.

The rest of this paper is organized as follows: Section 2 presents details of our approach. Section 3 reports experimental results. Section 4 introduces the related work. Section 5 concludes our work.

2 Methodology

While a corpus presents distributional properties of words, definitions provide semantic information of word senses in a compositional way. Therefore, we believe that we can compute word sense embeddings from definitions. We choose to use recurrent neural networks to model semantic compositionality because RNN-based models have been shown to be able to model semantic compositionality in many tasks, such as neural machine translation (Kalchbrenner and Blunsom, 2013; Bahdanau et al., 2014; Hermann et al., 2015), text entailment recognition (Rocktäschel et al., 2015) etc.

2.1 Definition Understanding Model

A word sense definition is a word sequence: \( \{x_1, x_2, ..., x_n\} \). As Figure 1 shows, RNN models take word embeddings of the words in definitions one by one and updates its internal memory according to its computation unit. The output of the RNN at the last word of the definition is assumed to contain the semantics of the definition. Hence we map \( h_n \) to sense embedding space with a transformation matrix:

\[
\tilde{e}_{ws} = W_h h_n + b_h
\]  

where \( W_h \) is the transformation matrix, \( b_h \) is the bias term and \( \tilde{e}_{ws} \) is the sense embedding computed by the definition understanding model.

The specific RNN model can be a standard RNN model, Gated Recurrent Unit (GRU) (Chung et al., 2014) or Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). Comparing with standard RNNs, LSTMs and GRUs can hold long-term information, i.e., they can alleviate the gradient vanishing and information-forgetting problem associated with the standard RNN for long sequences (Hochreiter and Schmidhuber, 1997; Chung et al., 2014).

2.2 Training Definition Understanding Model with definitions of Monosemous Words

Having determined the model structure, the challenge is how to train the RNN-based definition understanding model. Since the contexts of a monosemous word are associated with its only word sense, we assume its word sense embedding is similar to its word embedding. So we initialize sense embeddings with word embeddings for monosemous words and thus we can train the RNN-based model parameters with sense embeddings of monosemous words as target and their definitions as inputs. The used word
embeddings are trained using Skip-gram (Mikolov et al., 2013b) on a corpus. The word embeddings are also used to represent words in definitions as input to the RNN-based model. The word embeddings are kept fixed in the whole training process. As word embeddings trained on a corpus provide distributional properties of the words, our approach provides a supervised training for model parameters by combining distributional and compositional properties of word senses. The objective function of this training step is

$$J_1 = - \sum_{w \in V_{\text{mono}}} \cos(e_{ws}, \tilde{e}_{ws})$$

(2)

where $V_{\text{mono}}$ is the set of monosemous words and $e_{ws}$ is the initialized sense embedding of the monosemous word $w$ and $e_{ws}$ is the word sense embedding produced by our RNN-based definition understanding model. With this objective function, we train $e_{ws}$ to be similar to $e_{ws}$. Both the sense embeddings and word embeddings used in definitions are fixed in this step, i.e., we only train model parameters in this step.

2.3 Word Sense Embedding Learning

We have trained RNN-based definition understanding model with sense embeddings of monosemous words and their definitions in the previous step. However, we still haven’t use definitions of senses of multi-sense words. Since the word embeddings are trained according to the co-occurrences from a corpus, the word embedding of a multi-sense word usually represent the most common sense better and this is shown by our nearest neighbor evaluation in Section 3.2. So it is not appropriate to initialize all sense embeddings of a multi-sense word with its word embedding. Following Chen et al. (2014), we initialize each sense embedding of a multi-sense word with the average word embeddings of some candidate words from the sense definition. The candidate words include nouns, verbs, adjectives and adverbs and should have a similarity with the original word larger than a threshold $\delta$. Although these sense embeddings are simply initialized, we assume they still contain meaningful semantic information for training the definition understanding model and they will be also tuned by our model. Comparing with the last step, we still optimize the cosine similarities between embeddings produced by the RNN-based definition understanding model and the initialized word sense embeddings, but in this step we use definitions of both monosemous words and multi-sense words and update all sense embeddings jointly including sense embeddings of monosemous words because we think some of them are of low quality if the words are of low frequency in the corpus their word embeddings are trained. The objective function is as follows:

$$J_2 = - \sum_{w \in V} \sum_{s \in S_w} \cos(e_{ws}, \tilde{e}_{ws})$$

(3)

where $V$ is the whole word set and $S_w$ is the set of word senses of word $w$. To sum up, we make word sense embeddings and RNN-based definition understanding model train each other in this step.

2.4 Training with Word Sense Embeddings to Represent Words in Definitions

In the previous two steps, we train the definition understanding model and learn sense embeddings jointly using word embeddings to represent words in definitions. However, some words in definitions are multi-sense words and we assume sense embeddings are better to deal with these occasions, we use sense embeddings to represent words in definitions in this step.

To this end, we perform WSD for the words in definitions. We apply S2C (simple to complex) strategy described in (Chen et al., 2014). We identify their senses for words with less senses first and then for words with more senses. We compute the cosine similarity between each sense embedding of a word with its context embedding and choose the sense with greatest cosine similarity with the context embedding as the sense of the word. The context embedding is the average embedding of some other words in definitions. These words include nouns, verbs, adjectives and adverbs. We use the sense embeddings of those words whose senses have been identified and use the word embeddings of the rest words.

The objective function is the same as the previous step, but we use sense embeddings to represent words in definitions in this step and their sense embeddings are updated for the optimization of the objective function.
Table 1: Nearest neighbors based on cosine similarity between word embeddings or sense embeddings.

| Center Word/Sense | Nearest Neighbors |
|-------------------|-------------------|
| bank              | ATM machines, Iberiabank, automated teller machines |
| bank1             | financial, deposit, ATMs |
| bank2             | riverbank, water, slope |
| star              | matinee idol, singer, superstar |
| star1             | asteroid, celestial, supernova |
| star2             | legend, standout, footballer |
| pretty            | wonderfully, unbelievably, nice |
| pretty1           | remarkably, extremely, obviously |
| pretty2           | beauteous, dainty, lovely |

3 Experiments

We present qualitative evaluations and quantitative evaluations in this section. To show our word sense embeddings capture the semantics of word senses, we present the nearest neighbors of a word sense based on cosine similarity between the embedding of the center word sense and embeddings of other word senses. Besides, to show our model can actually understand a definition or a description we made up, we present the most matched word senses for a given description according to our RNN-based model. In our quantitative evaluations, we evaluate the sense embeddings on word similarity tasks and a word sense disambiguation task.

3.1 Setup

The lexical ontology we use is WordNet 3.0. We choose the publicly released 300 dimensional vectors trained with Skip-gram (Mikolov et al., 2013b) on part of Google News dataset (about 100B words) as word embeddings used in our approach. We also take the word embeddings as our baseline. Learning from the practice of (Chen et al., 2014), the similarity threshold \( \delta \) we used to initialize word sense embeddings of multi-sense words is set to be 0. We randomly initialize model parameters within (-0.012, 0.012) except that bias terms are initialized as zero vectors. We adopt Adadelta (Zeiler, 2012) with mini-batch to minimize our objective functions and set the initial learning rate to be 0.012.

3.2 Qualitative Evaluations

To illustrate the quality of our word sense embeddings, we show the nearest neighbors of words and of their senses in Table 1. The nearest neighbors of words are computed using word embeddings. The nearest neighbors of word senses are computed using word sense embeddings trained with our definition understanding model which uses GRU as its specific RNN. We leave out the sense numbers of nearest senses because the numbers are meaningless to be presented here. As can be seen, the nearest neighbors of the center words are normally associated with the most common sense of the word. Whereas, the nearest neighbors of word senses are associated with the corresponding sense of the word. Besides, some nearest neighbours (e.g., ”supernova”, ”dainty”) are rare to be seen in corpus. Therefore it shows the sense embeddings of rare words are even meaningful.

Although we train RNN-based definition understanding model to map a definition to its sense embedding, we will show the RNN-based definition understanding model can also understand descriptions we made up. We compute the cosine similarities between the model produced embedding with all word sense embeddings to find those most matched. We still choose GRU as the specific RNN. Table 2 shows the most matched word senses to the given descriptions. The descriptions in the upper subfield are definitions from WordNet and those in the under subfield are made up by us. Most of the predicted words match the meaning the descriptions convey and even those that don’t exactly match (e.g., ”gullible”),

---

1 http://wordnet.princeton.edu/
2 https://code.google.com/archive/p/word2vec/
Table 2: Using our definition understanding model to find the most matched word senses for descriptions. "falsifiable" are semantically relevant. The predicted words of the descriptions we make up are coincident with the descriptions. That illustrates our definition understanding model is effective to understand natural language.

3.3 Quantitative Evaluations

3.3.1 Word Similarity Evaluation on WordSim-353

WordSim-353 dataset (Finkelstein et al., 2001) consists of 353 pairs of nouns which are associated with human judgments on their similarities without context information. The evaluation metrics on this dataset is the Spearman’s rank correlation coefficient $\rho$ between the average human score and the cosine similarity scores predicted by the system.

Following Reisinger and Mooney (2010) [Huang et al., 2012], we use weighted average of cosine similarities between each possible word sense pair as the similarity of the two words. Since there is no context provided, the weights can be uniformly distributed which is adopted by Reisinger and Mooney (2010) [Huang et al., 2012] or determined by word sense frequency in the training set to compute weight which is adopted by Iacobacci et al. (2015). We choose to take the weights uniformly distributed. The following equation describe the weighted strategy:

$$\text{WeiSim}(w, w') = \sum_{i}^{n_1} \sum_{j}^{n_2} p(s_i|w)p(s_j|w') \cos(e_{w_{s_i}}, e_{w'_{s_j}})$$

where $w$ and $w'$ are the two given words, $n_1$ and $n_2$ are the number of senses of the two words and $p(s_i|w)$ and $p(s_j|w')$ are the normalized weights to use these senses to compute similarity, $e_{w_{s_i}}$ and $e_{w'_{s_j}}$ are sense embeddings.

Table 3 shows our results compared to previous approaches. Reisinger and Mooney (2010) propose to cluster the contexts of each word into groups and make each cluster a distinct prototype vector. Huang et al. (2012) also use contexts to determine the number of senses of a word and use global context to improve word representations. Neelakantan et al. (2014) extend Skip-gram (Mikolov et al., 2013b) to learn multiple embeddings per word. Wu and Giles (2015) cluster word senses and learn word sense embeddings from related Wikipedia concepts. Iacobacci et al. (2015) use BabelNet (Navigli and Ponzetto, 2012) as the word sense inventory and apply WSD to a corpus before they train word sense embeddings with Continuous Bag of Words (CBOW) architecture (Mikolov et al., 2013a).

As can be seen, our approach achieves significant improvement over the original word embeddings we use. Most improvements come from the step we train word sense embeddings with our RNN-based models when the definitions are represented still by word embeddings. We achieve further significant improvements when we continue jointly train the model and learn sense embeddings using sense embeddings trained in the previous step to represent words in definitions. It can be seen as an application of sense embeddings in a natural language understanding task, so is also illustrates our sense embeddings are better than word embeddings for natural language understanding from the perspective of real-world natural language understanding tasks. LSTM and GRU present comparable performances to be used as
Table 3: Performances on WordSim-353. The bottom subfield shows the performance of different settings of our system. SG represents just using word embeddings we acquired. Def (Word) represents the step in which we use word embeddings to represent words in definitions to train model and sense embeddings. Def (Sense) represents the step in which we use sense embeddings to represent words in definitions to train model and sense embeddings. * indicates statistical significant differences of t-test between performances of SG(100) and SG (100B)+Def (Word). ** indicates statistical significant differences of t-test between performances of SG (100B)+Def (Word) and SG (100B)+Def (Word)+Def (Sense) with the same RNN model.

| System | ρ × 100 |
|--------|---------|
| Reisinger and Mooney (2010) tf-idf (Wiki2.05B) | 76.0 |
| Huang et al. (2012) (0.99B) | 71.3 |
| Neelakantan et al. (2014) (0.99B) | 71.2 |
| Wu and Giles (2015) | 73.9 |
| Iacobacci et al. (2015) | 77.9 |
| SG (100B) | 66.5 |
| SG (100B)+Def (Word) (standard RNN) | 67.5(+1.0)* |
| SG (100B)+Def (Word)+Def (Sense) (standard RNN) | 68.2(+0.7)** |
| SG (100B)+Def (Word) (LSTM) | 73.9(+7.4)* |
| SG (100B)+Def (Word)+Def (Sense) (LSTM) | 74.8(+0.9)** |
| SG (100B)+Def (Word) (GRU) | 73.7(+7.2)* |
| SG (100B)+Def (Word)+Def (Sense) (GRU) | 74.5(+0.8)** |

3.3.2 Word Similarity Evaluation on Stanford’s Contextual Word Similarities

Since we need a context to determine the sense of a word when we use the sense embeddings in real-world tasks and evaluation on context-free word similarity datasets does not allow us to determine the sense, it cannot totally reveal the quality of our sense embeddings. Stanford’s Contextual Word Similarities (SCWS) (Huang et al., 2012) is a data set which provides the contexts of the target words. The way we determine the sense of the target words is the same S2C strategy we described in Section 2.4. Determining the senses of the target words, we compute cosine similarity of their sense embeddings as their similarity. The evaluation metrics is also the Spearman’s rank correlation coefficient ρ between the average human rating and the cosine similarity scores given by our approach.

Table 4 shows our results compared to previous approaches. Besides what we have mentioned, Chen et al. (2014) use WordNet to acquire number of senses of words and use definitions just to initialize sense embeddings and then train sense embeddings on a corpus processed with WSD model. Tian et al. (2014) model word polysemy from a probabilistic perspective and combine it with Skip-Gram (Mikolov et al., 2013b) model. Liu et al. (2015) incorporate topic models into word sense embedding learning. Li et al. (2015) use Chinese Restaurant Processes to determine the sense of a word and learn the sense embeddings jointly.

As can be seen, the improvements of each training step of our approach are in accordance with the results in WordSim-353 evaluation. LSTM and GRU also present much more improvements than standard RNN. Our proposed approach present high overall performance on both word similarity tasks. That illustrates the word sense embeddings indeed capture the semantics of word senses. Strictly speaking, the comparison between different approaches are not totally fair because the resources different approaches use are different.
| System                                           | \(\rho \times 100\) |
|-------------------------------------------------|----------------------|
| Huang et al. (2012) (0.99B)                     | 65.7                 |
| Chen et al. (2014) (1B)+WordNet                 | 68.9                 |
| Tian et al. (2014) (0.99B)                      | 65.4                 |
| Neelakantan et al. (2014) (0.99B)               | 69.3                 |
| Li et al. (2013) (120B)                         | 69.7                 |
| Liu et al. (2015) (0.99B)                       | 68.1                 |
| Wu and Giles (2015)                             | 66.4                 |
| Iacobacci et al. (2015)                         | 62.4                 |
| **SG (100B)**                                   | 64.4                 |
| **SG (100B)+Def (Word) (standard RNN)**         | 66.1 (+1.7)*         |
| **SG (100B)+Def (Word)+Def (Sense) (standard RNN)** | 66.8 (+0.7)**       |
| **SG (100B)+Def (Word) (LSTM)**                 | 68.8 (+4.4)*         |
| **SG (100B)+Def (Word)+Def (Sense) (LSTM)**     | 69.4 (+0.6)**        |
| **SG (100B)+Def (Word) (GRU)**                  | 68.9 (+4.5)*         |
| **SG (100B)+Def (Word)+Def (Sense) (GRU)**      | **69.5 (+0.6)** **   |

Table 4: Performances for our system and other proposed approaches on SCWS dataset.

| System                                           | F1        |
|-------------------------------------------------|-----------|
| Random                                          | 62.7      |
| Chen et al. (2014) (1B)+WordNet                 | 75.8      |
| **SG (100B)+Def (Word) (standard RNN)**         | 69.3      |
| **SG (100B)+Def (Word)+Def (Sense) (standard RNN)** | 70.2 (+0.9)** |
| **SG (100B)+Def (Word) (LSTM)**                 | 75.5      |
| **SG (100B)+Def (Word)+Def (Sense) (LSTM)**     | **76.2 (+0.7)** ** |
| **SG (100B)+Def (Word) (GRU)**                  | 75.6      |
| **SG (100B)+Def (Word)+Def (Sense) (GRU)**      | **76.2 (+0.6)** ** |

Table 5: Performances on Semeval-2007 coarse-grained all-words WSD task.

### 3.3.3 Word Sense Disambiguation Evaluation

If our word sense embeddings are able to capture the differences between senses of a word, the embeddings should be able to be applied in word sense disambiguation tasks. In Semeval-2007 coarse-grained all-words WSD task (Navigli et al., 2007), WordNet is used as the word sense inventory. But the evaluation of word sense disambiguation result is on a coarser-grained version of the WordNet sense inventory and those word senses which are hard to disambiguate even for humans are clustered into one class. The version of WordNet this task use is 2.1, but we learn our word sense embeddings with WordNet 3.0. We use the sense map[^3] between the two versions provided by the developers to address this issue. To compare the effectiveness of our word sense embedding on this task with previous work, following Chen et al. (2014), we still adopt the S2C strategy we described in Section 2.4 for disambiguating word sense. We also show the result produced by randomly choosing the sense of words according to (Chen et al., 2014).

The results are shown in Table 5. After we train our model and word sense embeddings using sense embeddings to represent words in definitions, our approach outperforms Chen et al. (2014) on this task. It illustrates that our sense embeddings can actually distinguish different senses of a word and our approach can actually learn the semantics of senses from definitions.

### 4 Related Work

Early word embedding learning approaches learn one embedding per word. Skip-gram (Mikolov et al., 2013b) and Glove (Pennington et al., 2014) are the most prevalent models of this kind.

[^3]: https://wordnet.princeton.edu/man/sensemap.5WN.html
Both of them use context information extracted from an unannotated corpus to learn word embeddings.

Since one embedding for each word sense are suggested to be better than a single embedding for a word, many word sense embedding learning approaches have been proposed (Reisinger and Mooney, 2010; Huang et al., 2012; Chen et al., 2014; Neelakantan et al., 2014; Tian et al., 2014; Iacobacci et al., 2015; Liu et al., 2015; Wu and Giles, 2015; Li and Jurafsky, 2015). Researchers tend to extend Skip-gram and Glove models to learn sense embeddings with WSI or WSD as a preliminary. Reisinger and Mooney (2010) propose to cluster the contexts of each word into groups and make each cluster a distinct prototype vector. Huang et al. (2012) determine the sense of a word by clustering the contexts and then apply it to neural language model with global context. Neelakantan et al. (2014) extend Skip-gram (Mikolov et al., 2013b) to a model which jointly performs word sense discrimination and embedding learning. Liu et al. (2015) associate words with topics and then extend Skip-gram (Mikolov et al., 2013b) to learn sense and topic embeddings. Wu and Giles (2015) propose to use Wikipedia concepts to cluster word senses and to learn sense-specific representation of words. Li et al. (2015) use Chinese Restaurant Processes to determine the sense of a word and learn the sense embedding jointly. Iacobacci et al. (2015) use BabelNet (Navigli and Ponzetto, 2012) as the word sense inventory and opt for Babelfy (Moro et al., 2014) to perform WSD on Wikipedia. Then they train word sense embeddings using CBOW architecture (Mikolov et al., 2013a) on the processed corpus. Chen et al. (2014) use WordNet as its lexical ontology to acquire numbers of word senses and use the average word embedding of words chosen from definitions as the initialization of sense embeddings. And then they do WSD on a corpus and train sense embeddings with a variant of Skip-gram on the corpus. Both of our approaches use words in definitions to initialize word sense embeddings, but after that their training still concentrates on the corpus but we train our model and word sense embeddings with definitions. The disadvantage to use a corpus processed by WSD or WSI may come from the unreliability of the processing results and since a corpus for embedding training are usually much larger in scale than the summation of all the definitions to get satisfied result, their approach inevitably consumes much more time on WSD and training.

5 Conclusion

In this paper, we propose to use RNN-based model to learn word sense embeddings from sense definitions. Our approach produces an effective natural language understanding model and word sense embeddings of high quality. Comparing with previous work training word sense embeddings on a corpus, our approach is less time-consuming and better for rare word senses. Experimental results show our word sense embeddings are of high quality.

References

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

[Chen et al.2014] Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In EMNLP.

[Chung et al.2014] Junyoung Chung, aglar Gülehrle, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555.

[Finkelstein et al.2001] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. ACM Trans. Inf. Syst., 20:116–131.

[Guo et al.2014] Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In COLING.
[Hermann et al.2015] Karl Moritz Hermann, Tomáš Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9:1735–1780.

[Huang et al.2012] Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[Iacobacci et al.2015] Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Senseemb: Learning sense embeddings for word and relational similarity. In *ACL*.

[Kalchbrenner and Blunsom2013] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

[Li and Jurafsky2015] Jiwei Li and Daniel Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *EMNLP*.

[Liu et al.2015] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI*.

[Mikolov et al.2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[Mikolov et al.2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

[Miller1992] George A. Miller. 1992. Wordnet: A lexical database for english. *Commun. ACM*, 38:39–41.

[Moro et al.2014] Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *TACL*, 2:231–244.

[Navigli and Ponzetto2012] Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250.

[Navigli et al.2007] Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. 2007. Semeval-2007 task 07: Coarse-grained english all-words task.

[Neelakantan et al.2014] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

[Reisinger and Mooney2010] Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *NAACL*.

[Rocktäschel et al.2015] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kociský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *CoRR*, abs/1509.06664.

[Suster et al.2016] Simon Suster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. *CoRR*, abs/1603.09128.

[Tian et al.2014] Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*.

[Wu and Giles2015] Zhaohui Wu and C Lee Giles. 2015. Sense-aware semantic analysis: A multi-prototype word representation model using wikipedia. In *AAAI*, pages 2188–2194.

[Zeiler2012] Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701.