Different Contexts Lead to Different Word Embeddings

Wenpeng Hu, Jiajun Zhang*, Nan Zheng*
Institute of Automation, Chinese Academy of Sciences, China
{wenpeng.hu,jjzhang}@nlpr.ia.ac.cn
nan.zheng@ia.ac.cn

Abstract

Recent work for learning word representations has applied successfully to many NLP applications, such as sentiment analysis and question answering. However, most of these models assume a single vector per word type without considering polysemy and homonymy. In this paper, we present an extension to the CBOW model which not only improves the quality of embeddings but also makes embeddings suitable for polysemy. It differs from most of the related work in that it learns one semantic center embedding and one context bias instead of training multiple embeddings per word type. Different context leads to different bias which is defined as the weighted average embeddings of local context. Experimental results on similarity task and analogy task show that the word representations learned by the proposed method outperform the competitive baselines.

1 Introduction

Different from traditional one-hot sparse vector representation, word embeddings are dense and low-dimensional. Due to natural advantage in word similarity computation, word embeddings are useful in a variety of applications, such as information retrieval (Uddin et al., 2013; Ganguly et al., 2015), sentiment analysis (Santos et al., 2014; Nguyen et al., 2015), question answering (Tellex et al., 2003) and parsing (Socher et al., 2013). Researchers learn word embeddings in various ways. Matrix Factorization methods for generating dense word embeddings have been used for years (Lund et al., 1996). While in recent years, neural network and deep learning have become popular approaches for learning word embeddings since Bengio et al. (2003) introduced feed forward neural network into traditional n-gram language models. For example, Collobert and Weston proposed a new objective function to learn word embeddings and improved word embeddings' quality (Collobert et al., 2008). Huang et al. (2012) presented a new neural network architecture which incorporated both local and global document context, and offered an impressive result. The word2vec toolkit developed by (Mikolov et al., 2013a; Mikolov et al., 2013b) implemented both Skip-gram and CBOW models which could provide high-quality word embeddings.

In spite of many successful applications, most word embeddings have a common problem that each word is represented by a single vector, subsequently ignoring polysemy. For example, the word *bank* cannot have high cosine similarity with the word *river* and *money* at the same time since these two words are so dissimilar and the single vector representation of word *bank* cannot express two different meanings. Thus, several multi-prototype models have been proposed to alleviate the problem caused by the polysemy and homonymy. Guo et al. (2014) took advantages of bilingual resources and affinity propagation clustering algorithm to learn multiple embeddings corresponding with multiple word senses. Due to the limitation of bilingual resources, they couldn't train their model on large scale corpus. Huang et al. (2012) pre-clustered the contexts of a word into K classes, and then learned K embeddings per word. K was a predefined value that would make matters confusion because different words usually had different number of senses. Neelakantan et al. (2014) shifted clusters into the training progress and

* The authors contributed equally to this paper

proposed a non-parametric clustering model which could dynamically generate new cluster center by similarity threshold λ and thus could learn different number of embeddings per word type. Although they solved the problem caused by predefined K cluster centers, hyper-parameter λ was still hard to be defined, because the similarity thresholds for different words might be different. Moreover, they were sensitive to noise. Once a noise point happened, a new cluster center would be created. Therefore it would lead to mistakes due to no actions were taken to remove these incorrect clusters. Yang et al. (2016) proposed a supervised fine tuning framework to transform the existing single-prototype word embeddings into multi-prototype word embeddings based on lexical semantic resources. It was a good idea to obtain multi-prototype word embeddings, but it depended on the existing embeddings which had lost some information during the training process.

To the best of our knowledge, most of the prior work deal with the polysemy problem by learning multiple embeddings per word type. In this paper, we present a novel approach to take the polysemy phenomena into consideration. It is based on the assumption that the same word in different contexts has different meanings and so the polysemy problem appears. Traditional single word embedding is approximately the weighted mean of its different contextual semantics. Polysemic words owning the same appearance may mean that they share a common intrinsic quality, and the differences among them are reflected by different contexts. In other words, the different contexts of a word exert different influences, and the given context is a good indication of the direction of the word meaning. Therefore, we propose a method to model the intrinsic quality of words and obtain the influences affected by contexts.

In this paper, we make the following contributions:

- The composition and structure of our word embeddings are the word center adding the offset induced by the word contexts, which make it flexible to deal with the polysemy problem.
- There is no need to cluster before or during training process, which removes the inaccuracy caused by clustering. Each component of our word embeddings can be updated or trained thousands of times during the training process which makes it robust and less susceptible to outliers.
- We quantify the influencing degree of the local contexts by training weighted parameters. In this way, the first part in our word embeddings named word center can be trained more accuracy and less ambiguous.

2 Background: CBOW (Continuous Bag-of-Words Model)

As a popular toolkit provided by Mikolov et al., word2vec[1] has gained lots of attractions in recent years. Two models CBOW and Skip-gram (Mikolov et al., 2013a; Mikolov et al., 2013b) are used, which make it possible to train on more than 100 billion words in one day and generate high-quality word embeddings. Word2vec provides two frameworks for these two models. One is based on Hierarchical Softmax and the other is based on Negative Sampling. According to (Mikolov et al., 2013b), Negative sampling (NEG) is a simplification of Noise Contrastive Estimation (NCE) (Gutmann et al., 2012), which can improve both of the training speed and the quality of word embeddings. So we choose Negative sampling to optimize our model.

The CBOW model uses the contexts of word w to predict w. Thus, for the given contexts, word w is a positive sample and other words are negative samples. In the CBOW model, $v(w) \in R^d$ is the word embedding of word $w \in W$, where W is the word vocabulary and d is the dimension of the embedding. $C(w) \in R^d$ is the average embedding of the context for word w.

Given the context of word w, the probability that a word u can be observed is given by,

$$P(u|Context(w)) = \sigma(C(w)^T\theta^u) = \frac{1}{1+e^{-C(w)^T\theta^u}} \tag{1}$$

where θ^u are training parameters of word u according to the source code of Word2vec. For the positive samples where $u = w$, the CBOW model is maximizing formula (1), while for the negative samples

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

$u \in NEG(w)$ the model will minimize it which can be done as maximize $1 - \sigma(C(w)^T \theta^u)$. $NEG(w)$ is the sets of negative samples for word w. So rewrite formula (1), given a samples u, CBOW model will maximize:

$$g(w) = \prod_{u \in w \cup NEG(w)} p(u|Context(w)) \tag{2}$$

where

$$p(u|Context(w)) = \begin{cases} \sigma(C(w)^T \theta^u), & L^w(u) = 1; \\ 1 - \sigma(C(w)^T \theta^u), & L^w(u) = 0, \end{cases}$$

$$L^w(u) = \begin{cases} 1, & u = w; \\ 0, & u \neq w, \end{cases}$$

Then the whole expression of $g(u)$ can be written as:

$$g(w) = \prod_{u \in w \cup NEG(w)} [\sigma(C(w)^T \theta^u)]^{L^w(u)} \cdot [1 - \sigma(C(w)^T \theta^u)]^{1 - L^w(u)}$$

Given a training corpus $\mathcal{C}$, the word embeddings are learned by maximizing the following objective function:

$$G = \prod_{w \in \mathcal{C}} g(w) \tag{3}$$

For compute easily, the objective function take the $\log$ and switchs from product to sum:

$$\begin{aligned} \mathcal{L} = \log G &= \log \prod_{w \in \mathcal{C}} g(w) = \sum_{w \in \mathcal{C}} \log\ g(w) \\ &= \sum_{w \in \mathcal{C}} \sum_{u \in \{w\} \cup NEG(w)} \mathcal{L}(w, u) \end{aligned}$$

where

$$\mathcal{L}(w, u) = L_w(u) \cdot \log[\ \sigma(\ C_w^T\ \theta^u)] + [1 - L^w(u)] \cdot \log[1 - \sigma(C_w^T\ \theta^u)] \tag{4}$$

For the negative sampling method, Mikolov et al. found that the powered unigram distribution $U(m)^{3/4}$ outperformed significantly the unigram and the uniform distributions. The distribution that the noisy contextual words are randomly sampled by is as following:

$$P(w) = \frac{p_{unigram}(w)^{3/4}}{Z} \tag{5}$$

where $p_{unigram}(w)^{3/4}$ is the number of occurrences of the words and Z is the normalization constant.

3 The RLC (Reuse Local Context) Model

3.1 The Embedding Structure of RLC

Different from traditional one-hot vector representation approach, each dimension of word embedding represents a latent feature of the word, mentioned by (Guo et al., 2014). And the word analogical task introduced by (Mikolov et al., 2013a) means that the influence between words can be simply computed by addition and subtraction. Because of these, we can assume that the dimensions between a word embedding are relatively independent which means there are no cross influence between them. Then we introduce a weighted parameter for each word as a filter to control the influence degree given by the context of the word. And the center of the word embedding will be shifted after adding the contextual influences to word embeddings.

In our approach, the word embedding structure of word w can be written as following:

$$[Xe_w; Xm_w] \tag{6}$$

where $Xe_w \in R^d$ is the word embedding center point of word w, $Xm_w \in R^d$ is the weighted parameters.

Using this structure can obtain different word embeddings according to different contexts. If there are no contextual words, the center point of word embedding Xe_w can be used instead. Considering a context and a word sampled from it, it is impossible for the context to have negative influence to the word. So we use the sigmoid function to limit Xm_w within the range of 0 to 1. And the combination formula of word embedding for w is as follows:

$$X_w = Xe_w + \sigma(Xm_w) \cdot \frac{1}{m} \sum_{i=1}^{m} Xe^w{}_i \tag{7}$$

where m represents the number of words in the context of w; $Xe^w{}_i$ is the *i*th word in the context of w.

3.2 The Algorithm of RLC

We introduce our algorithms into CBOW model, named RLC on CBOW (RLCC). Then, we build our objective function. Unfortunately, the probability computing method that used by CBOW model is not suitable for our approach according to formula (1-3). Because they calculate the probability of word u given context $Context(w)$ using inner product of $C(w)$ and θ^u. While, θ^u are training parameters corresponding to the sample word u, which are of no use after training. So we cannot combine the influences of the given context with parameters u. If we combine every word in $Context(w)$ by formula (7), then the computational complexity of our model will be heavily increased. But we find that the function of θ^u is semantic representation for word u and it is also a kind of word embedding for word u which can be replaced by the true word embedding. So we update the probability $p(u|Context(w))$ by:

$$p(u|Context(w)) = \begin{cases} \sigma(C(w)^T X_u), & L^w(u) = 1; \\ 1 - \sigma(C(w)^T X_u), & L^w(u) = 0, \end{cases} \tag{8}$$

where

$$X_u = Xe_u + \sigma(Xm_u) * Xc$$

$$Xc = \frac{1}{m} \sum_{i=1}^{m} Xe^w{}_i = C(w) \tag{9}$$

where "$*$" denotes the product of corresponding vector components, Xc denotes the average embeddings for the current context of word w.

Negative sampling maximizes the probability of positive samples given the context and at the same time minimizes the negative ones. But there are some problems for our approach to simply use Negative sampling. In the training process, negative samples are treated completely irrelevant with the given context, and the probability that they are observed would be minimized. This is beneficial for the model to differentiate data from noise. While from the right part in formula (7) recorded as $X_{influence} = \sigma(Xm_w) \cdot \frac{1}{m} \sum_{i=1}^{m} Xe^w{}_i$ we can see that minimizing observed probability will be achieved by decreasing Xm_w, and this satisfies our goal. Unfortunately, the decreased value of $\sigma(Xm_w)$ for every iteration is $||C(w)||_2^2$ which has a mean greater than zero and should not be considered as probability. Meanwhile, according to the paper of (Mikolov et al., 2013b), we should ensure the effect that the number of negative samples should be in the range of 5-20 for small training datasets and in the range of 2-5 for large datasets. That means the number of negative samples is at least two times bigger than positive samples which makes matter worse. So we introduce extra learning rates to solve the problem:

$$\eta = L^w(u) * \alpha + \beta \tag{10}$$

where α and β are hyperparameters.

The influences of context to positive samples must be positive, so we introduce the sigmoid function to limit Xm_u within the range of 0-1. But it is not reasonable for negative samples, because the influences of context to negative samples could be negative. Meanwhile, Xc given as in formula (9) still has strong

correlation with the context embedding $C(w)$ after multiply by the weight $\sigma(Xm_u)$ which is harmful to the update of Xe_u in positive samples during the training process. We introduce new extra weighted parameters Tm during training process to solve the problem which can be seen as the weight for negative samples:

$$Xm_{u\_new} = (1-\gamma) \cdot Tm_u + \gamma \cdot \sigma(Xm_u) \tag{11}$$

where $\gamma = L^w(u) * (1-\lambda) + (1-L^w(u)) * \lambda$,$\lambda \in [0,1]$. For example, if we set $\lambda = 0$, then $\sigma(Xm_u)$ will be used as the weighted parameter when the sample is positive one.

In this case, formula (7) should be rewritten according to formula (11):

$$X_w = Xe_w + Xm_{w\_new} \cdot \frac{1}{m} \sum_{i=1}^{m} Xe^w{}_i \tag{12}$$

Combining the given objective function (3-4) and the above analysis, we can infer the update gradient in each iteration as follows and the pseudo-code is showing in the Algorithm 1.

$$\begin{aligned}
\frac{\partial \mathcal{L}(w,u)}{\partial Xe^w} &= [L^w(u) - \sigma(C(w)^T X_u)] * \frac{1}{m} * (\, X_u + C(w) * Xm_{u_new}) \\
\frac{\partial \mathcal{L}(w,u)}{\partial Xe_u} &= [L^w(u) - \sigma(C(w)^T X_u)] * C(w) \\
\frac{\partial \mathcal{L}(w,u)}{\partial Xm_u} &= [L^w(u) - \sigma(C(w)^T X_u)] * C(w) * Xc * \sigma(Xm_u) * (1 - \sigma(Xm_u)) * \eta * \gamma \\
\frac{\partial \mathcal{L}(w,u)}{\partial Tm_u} &= [L^w(u) - \sigma(C(w)^T X_u)] * C(w) * Xc * (1-\gamma)
\end{aligned} \tag{13}$$

Algorithm 1 : Training Algorithm of RLC model

Input : Corpus $\mathcal{C}$.

Initialize : $Xw \; and \; Tw, \forall \; w \in Vocab, Xw \in R^{2d}, Tw \in R^d$, randomly.

while next sentence in $(\mathcal{C})$ not null **do**

  $e = 0$.

  $C_w = \frac{1}{m} \sum_{u \in Context(w)} Xe_u$

  for $u = \{w\} \cup NEG(w)$ **do**

    $Xm_{u\_new} = (1-\gamma) \cdot Tm_u + \gamma \cdot \sigma(Xm_u)$

    $X_u = Xe_u + Xm_{u\_new} C_w$

    $q = \sigma(C_w^T X_u)$

    $g = \eta_1 (L^w(u) - q)$

    $e := e + g(\, X_u + C_w Xm_{u\_new})/m$

    $Xe_u := Xe_u + gC_w$

    $Xm_u := Xm_u + gC_w{}^2 \sigma(Xm_u)(1 - \sigma(Xm_u))\eta\gamma$

    $Tm_u := Tm_u + gC_w{}^2 (1-\gamma)$

  end for

  for $u \in Context(w)$ **do**

    $Xe_u := Te_u + e$

  end for

end while

---

4 Experiments

Following previous work (Huang et al., 2012; Neelakantan et al., 2014), we choose Wikipedia corpus[2] (Shaoul et al., 2010) snapshotted at April 2010 to train embeddings. The Wikipedia corpus contains a total of about 2 million articles and 990 million tokens. In all of our experiments we remain 220682 words as the vocabulary by removing all the words with less than 55 occurrences and set the maximum context window ($m/2$) to be 8 which means 8 words before and after the word occurrence. Our hyper-parameter values are selected by manual exploration of results when using a small corpus attached by Word2vec toolkit, named text8 which has 31893 words in the vocabulary and 16 million words in the corpus after removing the words with the occurrences less than 20. In RLCC we set $\lambda = 0.5,\ \alpha = 0.4,\ \beta = 0.6$. Also we train embeddings in different dimensions and compare our methods with many state-of-the-art models.

Table 1 shows the training time of our models on the small data set text8, compared with other models from previous work. All the methods are evaluated using a single-machine with 16 threads. We see that the training speed of our model is faster than Skip-gram model but is slower than CBOW model. This indicates that our model can be acceptable with respect to efficiency.

Model	**Time (minute second)**		
	dim-50	dim-100	dim-300
CBOW	55s	1m10s	2m33s
Skip-gram	5m35s	7m32s	18m57s
RLCC	1m52s	3m1s	7m14s

Table 1: Training Time Comparison

4.1 Word Similarity

The two datasets that we used to evaluate our word embeddings are as follows: the WordSim-353 (Finkelstein et al., 2001) dataset and the Contextual Word Similarities (SCWS) dataset (Huang et al., 2012). WordSim-353 is a standard dataset for evaluating word embeddings which consists of 353 pairs of word types. The similarity of each word pair is manually rated in a scale from 0 to 10 by 13 to 16 human judgements, and each pair receives an average score. But these scores are given without any information of the context which makes our embeddings hard to use. Fortunately, we are surprised to see that our embeddings outperform competitive baselines and even the state-of-the-art embeddings by only using the word center embeddings Xe.

Because of the absence of contextual information in the WordSim-353 dataset, Huang et al. (2012) developed Stanford's Contextual Word Similarities (SCWS) dataset which consists of 2003 word pairs and associated sentential contexts. So the SCWS dataset overcomes the issue caused by no contextual information in WordSim-353 and the models designed to deal with polysemous problems could have a good testbed. Most of those models train multiple embeddings per word type to tackle the polysemous problems and there are many approaches to measure the similarities of multi-prototype embeddings according to (Reisinger et al., 2010). Then we select the best results for each model to compare with our model.

Table 2 shows our results compared to previous methods on WordSim-353 dateset and Table 3 gives the results on SCWS dataset. All the models mentioned are trained in the same Wikipedia corpus snapshotted at April 2010. The scores Huang et al. we used in the table is using the word embeddings trained and provided by (Huang et al., 2012). MSSG and NP-MSSG are the best scores provided by (Neelakantan et al., 2014). We train Skip-gram and CBOW models using 10 negative samples and a context window size of 8 on the same corpus. RLCC* is our model, and the results shown in the table are gained by only using Xe in our models. Some results are not provided in these tables because we could not find the embeddings or results provided by the authors. From Table 2 we can see that our model is able

[2]http://nlp.stanford.edu/data/WestburyLab.wikicorp.201004.txt.bz2

Model	Different dimensions		$\rho \times 100$
	dim-50	dim-100	dim-300
Huang et al.	64.2	-	-
C&W	55.3	-	-
MSSG	63.2	-	70.9
NP-MSSG	62.4	-	68.6
CBOW	65.0	66.5	65.8
Skip-gram	68.6	71.7	72.7
RLCC*	**70.8**	**73.0**	**75.0**

Table 2: Experimental Results on the WordSim-353 dataset. The numbers in the table are Spearmans correlation recorded as $\rho \times 100$ between the model's similarities and human judgments. The best results according to each dimension are in bold face.

to learn more semantic word embeddings and noticeably improves upon previous models. We believe that the improvement is mainly attributed to the fact that the learned center word embedding Xe is less ambiguous.

Table 3 shows that our model gets the best result on both the 50-dimension embeddings and the 300-dimension embeddings. Obviously, our approach can better deal with polysemous problems than the traditional multi-prototype word embeddings.

Model	Dim	NUContext	UContext
C&W	50	57.0	-
CBOW	50	64.7	-
Skip-gram	50	63.4	-
Huang et al.	50	58.6	65.7
MSSG	50	62.1	66.9
NP-MSSG	50	62.3	66.1
RLCC	50	65.3	67.3
CBOW	100	65.8	-
Skip-gram	100	64.9	-
RLCC	100	66.0	68.4
CBOW	300	66.6	-
Skip-gram	300	66.7	-
MSSG	300	65.3	69.3
NP-MSSG	300	65.5	69.1
RLCC	300	**67.6**	**69.7**

Table 3: Experimental results on the SCWS dataset. "NUContext" is the abbreviation for "Not use the contextual information". For those multi-prototype embeddings, the $globalSim$ metric which is each word's global context vector ignoring the many senses is used. While in our model, we still use Xe for each word, ignoring the weighted parameters. "UContext" is the abbreviation for "Use the contextual information". In this time, for multi-prototype embeddings models, $avgSimC$ method is applied which weights the similarity by how well each sense fits the context at hand (Neelakantan et al., 2014). All of the best results for each dimension are marked by underline, while in the two situations of using context or not, they are marked by bold.

4.2 Word Analogies

The word analogy task is introduced by (Mikolov et al., 2013a). It is a comprehension test task which is designed to measure the quality of word embeddings. This task consists of questions like, "a is to b

as c is to ___ ?". The dataset we applied in this task is built by Mikolov et al., which contains five types of semantic questions and nine types of syntactic questions with 8869 and 10675 questions respectively. The questions, for example, "Tokyo is to Japan as Beijing is to ___ ?", are answered by finding the word whose embeddings are the most similar one with $W_{Japan} - W_{Tokyo} + W_{Beijing}$ under cosine similarity.

According to (Neelakantan et al., 2014), both MSSG and NP-MSSG models achieve 64% accuracy which exceed (Huang et al., 2012) but are worse than the Skip-gram model. So we compare our model with CBOW and Skip-gram models only in this task. The results shown on Table 4 demonstrate that our model outperforms both the CBOW and Skip-gram models in most words' analogy task especially in answering semantic questions. It means that our embeddings are less constrained by syntax and are more semantic.

Model	Dim	Sem.	Syn.	Tot.
CBOW	50	59.6	59.4	59.5
Skip-gram	50	52.5	55.9	54.6
RLCC*	50	66.3	60.1	62.5
CBOW	100	72.8	70.2	71.2
Skip-gram	100	67.8	69.2	68.6
RLCC*	100	80.0	70.8	74.3
CBOW	300	77.9	**75.4**	76.4
Skip-gram	300	83.1	73.4	77.2
RLCC*	300	**90.7**	71.8	**79.0**

Table 4: Experimental results of the word analogy task show as percent accuracy. We trained CBOW and Skip-gram using the same corpus as our model used, and the training parameters were also the same as we described before.

5 Related Work

In recent years, neural network and deep learning have become popular approaches for learning word embeddings, which make it possible to study dense and high quality word embeddings. Bengio et al. (2003) introduced feed forward neural network into traditional n-gram language models, which might be the foundation work for neural network language models(NNLM). In NNLM, words were represented by a low-dimensional vector and the parameters could be learned in unsupervised methods. Collobert et al. (2008) proposed a new objective function to learn word embeddings instead of the time consuming softmax layer presented in (Bengio et al., 2003) and much improved training speed. Mnih et al. (2007) reduced the computational complexity of the Bengio's model by replacing the softmax layer with a tree-structured probability distribution.

Mikolov et al. (2013a) and Mikolov et al. (2013b) removed the hidden layers of neural network and proposed log-liner neural language models named Skip-gram and CBOW. These two models extremely reduced the computational complexity and could train word embeddings on more than 100 billion words in one day with a single machine. With the help of negative sampling method, both of the two models could obtain state-of-the-art word embeddings.

In the previous work, several multi-prototype models have been proposed to alleviate the problem caused by the polysemy and homonym. Guo et al. (2014) took advantages of bilingual resources and affinity propagation clustering algorithm to learn multiple embeddings corresponding with multiple word senses, because a polysemous word in one language could not be exactly a polysemous word in another language. Huang et al. (2012) pre-clustered the corpus into specified classes, and relabeled the tokens into different classes, then learned specified numbers of embeddings per word type. The number of each word senses was predefined as a fixed value that would make matters confusion because the different words might have different number of senses. Neelakantan et al. (2014) shifted clusters into the training progress and proposed a non-parametric clustering model which could dynamically generate new clusters based on word meaning. Fine tuning was also a good idea to generate muti-prototype word embeddings.

Yang et al. (2016) proposed a supervised fine tuning framework to transform the existing single-prototype word embeddings into multi-prototype word embeddings based on lexical semantic resources.

6 Conclusion

In this paper, we present a novel model to reuse the local context and enhance word embeddings for both monosemous and polysemous words. The proposed model is an extension to CBOW and it is designed to embed a word as a word semantic center and a weighted parameter. Weighted parameter denotes the influences given by the contexts. Experimental results show that embeddings trained by our model outperform competitive baselines and even state-of-the-art embeddings. When we focus on polysemy, our approach could shift the embedding of polysemous word into the corresponding semantic space according to the given context. In the future, we plan to make our model more explainable.

Acknowledgements

This research work has been partially funded by the Natural Science Foundation of China under Grant No. 91520204 and No. 61501463.

References

Mohammed Nazim Uddin, Trong Hai Duong, Ngoc Thanh Nguyen, Xin-Min Qi and GeunSik Jo. 2013. *Semantic similarity measures for enhancing information retrieval in folksonomies*, volume 40:1645–1653. Expert Syst. Appl.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes and Gregory Marton. 2003. *Quantitative evaluation of passage retrieval algorithms for question answering*. SIGIR.

Debasis Ganguly, Dwaipayan Roy and Mandar Mitra and Gareth J. F. Jones. 2015. *Word Embedding based Generalized Language Model for Information Retrieval*. SIGIR.

Cícero Nogueira dos Santos and Maíra A. de C. Gatti. 2014. *Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts*. COLING.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. *PhraseRNN: Phrase Recursive Neural Network for Aspect-based Sentiment Analysis*. EMNLP.

Richard Socher, John Bauer, Christopher D. Manning, Andrew Y. Ng. 2013. *Parsing with Compositional Vector Grammars*. ACL.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent and Christian Janvin. 2003. *A Neural Probabilistic Language Model*, volume 3:1137–1155. Journal of Machine Learning Research.

Ronan Collobert and Jason Weston. 2008. *A unified architecture for natural language processing: deep neural networks with multitask learning*. ICML.

Eric H. Huang, Richard Socher, Christopher D. Manning and Andrew Y. Ng. 2012. *Improving Word Representations via Global Context and Multiple Word Prototypes*. ACL.

Tomas Mikolov, Kai Chen, Gregory S. Corrado and Jeffrey Dean. 2013a. *Efficient Estimation of Word Representations in Vector Space*. Workshop at International Conference on Learning Representations(ICLR).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado and Jeffrey Dean. 2013b. *Distributed Representations of Words and Phrases and their Compositionality*. Advances in Neural Information Processing Systems(NIPS).

Jiang Guo, Wanxiang Che, Haifeng Wang and Ting Liu. 2014. *Learning Sense-specific Word Embeddings By Exploiting Bilingual Resources*. COLING.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos and Andrew McCallum. 2014. *Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space*. EMNLP.

Xuefeng Yang and Kezhi Mao. 2016. *Learning multi-prototype word embedding from single-prototype word embedding with integrated knowledge*, volume 56:291–299. Expert Syst. Appl.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman and Eytan Ruppin. 2001. *Placing search in context: the concept revisited*, volume 12:116–131. ACM Trans. Inf. Syst.

Michael Gutmann and Aapo Hyvärinen. 2012. *Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics*, volume 13:307–361. Journal of Machine Learning Research.

Cyrus Shaoul and Chris Westury. 2010. *The Westbury lab wikipedia corpus*.

Joseph Reisinger and Raymond J. Mooney. 2010. *Multi-Prototype Vector-Space Models of Word Meaning*, . The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics.

Andriy Mnih and Geoffrey E. Hinton. 2007. *Three new graphical models for statistical language modelling*. International Conference on Machine learning(ICML).

Kevin Lund and Curt Burgess. 1996. *Producing high-dimensional semantic spaces from lexical co-occurrence*. volume 28:203–208. Behavior Research Methods, Instrumentation, and Computers.