Approximating Word Ranking and Negative Sampling for Word Embedding

Guibing Guo[#*], **Shichang Ouyang**[#*], **Fajie Yuan**[†*], **Xingwei Wang**[#]
[#]Northeastern University, China
[†]University of Glasgow, UK
{guogb,wangxw}@swc.neu.edu.cn, 1701282@stu.neu.edu.cn, f.yuan.1@research.gla.ac.uk

Abstract

CBOW (Continuous Bag-Of-Words) is one of the most commonly used techniques to generate word embeddings in various NLP tasks. However, it fails to reach the optimal performance due to uniform involvements of positive words and a simple sampling distribution of negative words. To resolve these issues, we propose *OptRank* to optimize word ranking and approximate negative sampling for bettering word embedding. Specifically, we first formalize word embedding as a ranking problem. Then, we weigh the positive words by their ranks such that highly ranked words have more importance, and adopt a dynamic sampling strategy to select informative negative words. In addition, an approximation method is designed to efficiently compute word ranks. Empirical experiments show that *OptRank* consistently outperforms its counterparts on a benchmark dataset with different sampling scales, especially when the sampled subset is small. The code and datasets can be obtained from $https://github.com/ouououououou/OptRank$

1 Introduction

Word embedding is a technique to represent each word by a dense vector, aiming to capture the word semantics in a low-rank latent space. It has been widely adopted in a variety of natural language processing (NLP) tasks, such as named entity recognition, sentiment analysis and question answering due to its compact representation and satisfying performance. Among many different methods such as Glove [Pennington *et al.*, 2014], a well-known approach to implement word embedding is the Continuous Bag-of-Words model [Mikolov *et al.*, 2013], or CBOW[1]. It predicts a target word given a set of contextual words, where the target word is labeled as positive and the others are classified as negative. However, it treats all positive words equally regardless of their negative words, which are sampled merely based on popularity. The relations between positive and negative words have not been well utilized. As a result, the issues of positive word weights and negative word sampling hinder the performance of word embedding in both word analogy and word similarity tasks.

Recently some approaches have been proposed in the literature to help resolve these issues. An adaptive sampler [Chen *et al.*, 2017] has been proposed to roughly select the negative words which have larger inner products with contextual words than positive words, but it does not take care of the issue of positive word weighing. The WordRank model [Ji *et al.*, 2015] proposes to treat the word embedding as a ranking problem. The similarity is computed between the contextual words and a positive word and then fed into a ranking function, the result of which is adopted as the weights of positive words. Unfortunately, this model ignores the importance of negative sampling and thus only partially solves the issues in question. To sum up, there is no existing work that has carefully taken into consideration both issues of the CBOW model, indicating the importance and value of our research work. In this paper, we propose a novel ranking model called *OptRank* that optimizes the word ranking and approximates negative sampling for better word embedding, handling both issues of CBOW in a unified model. We transform the word embedding as a ranking problem, and ensure that positive words are likely ranked higher than negative ones. Specifically, we provide a thorough analysis of the CBOW model from the viewpoint of ranking optimization, and discuss the disadvantages in terms of positive word ranking and negative word sampling. Then, we propose the OptRank model to put more weights on positive words that lie in the position (rank) of closeness to the contextual words, and effectively choose the negative words that may be ranked higher than positive ones during the learning process. In addition, an approximation approach is devised to efficiently compute word ranks, avoiding the expensive rank search from the whole word space. In this way, our approach can not only outperform other state-of-the-art approaches by a significant margin on small corpora, but also be very competitive when the datasets are large. The experimental results on word analogy and word similarity tasks also verify the effectiveness of our approach.

2 Analysis of the CBOW Model

In this section, we first briefly introduce the CBOW model and then discuss its two main issues.

[*]The first three authors contributed equally and share the co-first authorship.

[1]We are aware that there are two ways to optimize CBOWs, namely hierarchical softmax [Mnih and Hinton, 2009] and negative sampling. Since negative sampling often achieves better performance than hierarchical softmax [Mikolov *et al.*, 2013], hence hereafter we refer CBOW as to CBOW by negative sampling.

2.1 The CBOW Model

To facilitate discussion, we introduce a number of notations. For a given corpus W, there are a bag (set) of words denoted by $w_1, w_2, \ldots, w_n$, where n is the number of words. For each word w_i, it can be embedded by a dense vector v_i. The objective of word embeddings is to learn proper values for each embedding vector $v_i \in \mathbf{R}^d$, where d is the dimension of latent feature space.

CBOW [Mikolov *et al.*, 2013] takes the advantage of contextual words, i.e., the surrounding words in two T-sized windows: $C(w_i) = \{w_{i-T}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+T}\}$. For simplicity, we use symbol w_p to denote the target (positive) word, and use symbol c to simplify its contextual set $C(w_p)$. Hence, the problem of word embedding can be formulated as: given a set of word-context (w_p, c) training pairs, optimizing a binary classification objective function to distinguish positive words from the negative ones. In other words, it can accurately predict a proper target word w_p that best suits a given context while the rest of candidate words should be labeled as negative, denoted by N. Each word in N is called negative word, denoted by $w_n \in N$. To generate the negative training examples, CBOW adopts a popularity-based strategy to sample negative words proportional to their popularity.

Specifically, the contextual words are summarized as a single vector, denoted by $v_c = \frac{1}{|c|}\sum_{p\in c} v_p$. Let $p(w|c)$ be the probability that the predicted word is w given context c. It is computed as follows:

$$p(w|c) = \begin{cases} \sigma(v_c^\top v_w), & w = w_p; \\ 1 - \sigma(v_c^\top v_w), & w \in N; \end{cases} \quad (1)$$

where $\sigma(\cdot)$ is a sigmoid function, transferring the similarity between context and word vectors into a probability value. CBOW aims to maximize $p(w_p|c)$ for target word w_p and minimize $p(w_n|c)$ for negative words N in the meanwhile. As a result, for each training example (w, c), the objective function given as follows:

$$\mathcal{J}_{(w,c)} = p(w_p|c) \prod_{w_n \in N} p(w_n|c) \quad (2)$$

We take the log value of the above function and substitute the variables by Eq. 1, and thus rewrite the CBOW objective function as:

$$\mathcal{J} = \sum_{(w,c)} \Big\{ \log\big(\sigma(v_c^\top v_{w_p})\big) + \sum_{w_n \in N} \log\big(1 - \sigma(v_c^\top v_{w_n})\big) \Big\} \quad (3)$$

2.2 Analysis of Positive Word Ranking

The first main issue of CBOW is the lack of mechanism to ensure that positive words w_p will be always ranked higher than negative words w_n, i.e., to correctly capture the semantic meanings of a word with respect to its context. For instance, suppose we have a sentence "There is a complex relationship between France and Germany", where the word 'France' is used as our target word w_p and the others are contextual words represented by v_c. By applying the CBOW model, we may predict the target word by ranking all the words in the corpus according to their similarity with the context, i.e., the inner product of v_w and v_c. The resulting ranked word list is shown in Table 1.

sign	-	-	-	-	-	+	-
word	cat	cheap	dog	women	jump	France	like
score	4.0	3.1	3.0	2.6	2.5	1.7	0.8
rank	1	2	3	4	5	6	7

Table 1: The resulting ranked word list, where the rank of target word 'France' is 6 with a relevance score 1.7, and the other words are denoted with negative signs but some of them (e.g, cat, cheap) are ranked higher than 'France' with greater scores.

We can observe that although the target word 'France' is ranked relatively high, some other (noisy) words such as 'cat' and 'dog' are ranked much higher than 'France', indicating the poor performance of the current approach. The computed similarity scores clearly cannot reflect the true semantic relations between each word and the given context. In this case, these results are not acceptable and further training is required to acquire a better predictive model. Although it is simply an intuitive example, our empirical study verifies that many such cases occur during the model learning based on real datasets.

This example inspires us to devise a fine-grained metric to estimate the ranking relations so that the word semantics can be more effectively represented.

$$\begin{aligned} \mathcal{L}_{(w,c)} &= \sum_{p\in W} sign(p) \cdot \frac{score(p,c)}{\log_2(1 + rank(p,c))} \\ &= \sum_{p\in W} sign(p) \cdot \frac{v_c^\top v_p}{\log_2(1 + rank(p,c))} \end{aligned} \quad (4)$$

where $sign(p)$ is a sign function to indicate if a word p is the target word ('+') or not ('-'), and $rank(p, c)$ is a function to calculate the rank value of the word p.

Intuitively, a larger value of $\mathcal{L}_{(w,c)}$ means the quality of the ranked words list is higher. A straightforward strategy for this goal is to maximize the relevance score (inner product) for the positive word with the highest rank, and to minimize the inner products for the negative words. Thus, we deduce a general criterion for ranking optimization: for any $w_n \in N$, $v_c^\top v_{w_p} - v_c^\top v_{w_n} > 0$. To enhance the model generalization, it is also required to increase the difference between $v_c^\top v_{w_p}$ and $v_c^\top v_{w_n}$ as large as possible. Hence, we may conclude that the proper rule for ranking optimization is:

$$v_c^\top v_{w_p} - v_c^\top v_{w_n} > \epsilon, \quad \forall w_n \in N, \quad (5)$$

where $\epsilon > 0$ is a threshold to indicate how well the learned model can perform.

Therefore, although CBOW will separately increase the ranking scores of positive words and decrease those of negative words in each iteration, there is no guarantee that the estimated ranking scores between positive and negative word pairs can satisfy the requirements as given in Eq. 5. In this regard, CBOW can only achieve suboptimal performance.

2.3 Analysis of Negative Word Sampling

The second issue of CBOW is the negative word sampling strategy is solely based on word popularity, which has nothing

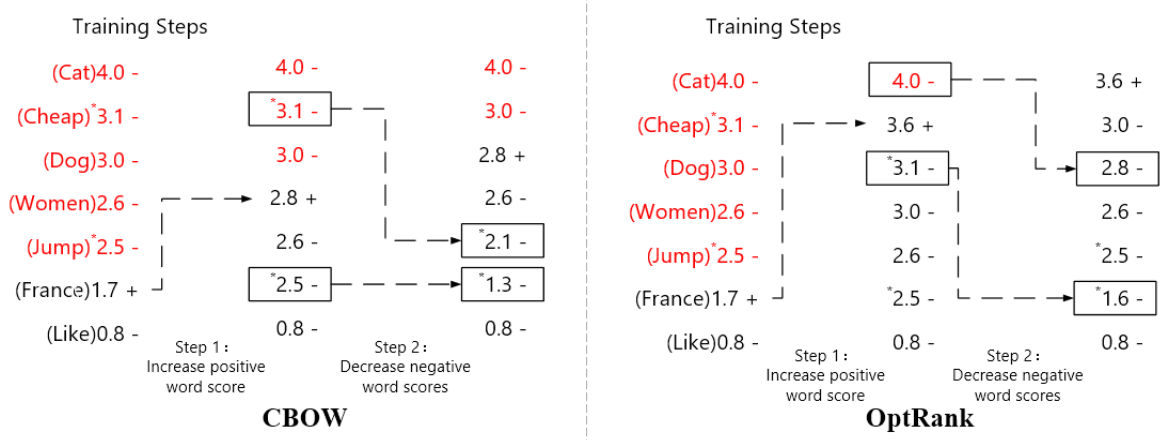

Figure 1: CBOW vs. OptRank on a specific word 'France' as mentioned in Table 1. The word denoted with symbol '*' means that it is a popular word in the corpus. And the red words are the negative words which have higher scores than the positive word. Both models first increase the ranking scores of positive words (step 1) and then decrease the ranking scores of negative words (step 2).

to do with the positive word. Without consideration the relations between positive and negative words, it is hard to make sure the optimization criterion of Eq. 5 will be met eventually. Next we take a closer look at several sampling strategies and analyze if they may meet the ranking requirements.

A straightforward approach perform negative sampling is to randomly select negative words from the whole corpus. It is easy-to-implement and efficient, but may ignore many important negative words that are ranked higher than positive words, due to the long tailed word distribution [Chen *et al.*, 2017]. Moreover, most randomly sampled negative words are not important for embedding learning because they are originally ranked lower than positive words. In this regard, the popularity-based sampling by CBOW can ensure more connections between positive and negative words, since popular words may appear frequently to build connections with lots of positive words.

A stronger solution is to purposefully choose the negative words that have potentially high similarity with positive words. An adaptive sampler [Chen *et al.*, 2017] was proposed to strategically select those negative words that have larger inner products with contextual words than positive words. Putting these example words into the training process will force the model to learn better to distinguish similar words.

3 The OptRank Model

This section aims to elaborate the formulation of our OptRank model for bettering word embedding, and an effective learning scheme to reduce the computational cost when estimating word ranks. Lastly, we discuss some practical issues and solutions when applying our approach in real datasets.

3.1 Model Formulation

In this paper, we regard the word embedding as a ranking problem as discussed in the previous section and represented by Eq. 4. To estimate the quality of a resulting word list for a given word-context (w_p, c) pair, it is required to define a ranking function $rank(w, c)$ (see Eq. 4). Specifically, for a positive word w_p, its rank value is the same as the number of words in the corpus that have a greater similarity (thus ranked higher) with the given context c, given by:

$$rank(w_p, c) = \sum_{w \in W} I(v_c^\top v_{w_p} < v_c^\top v_w + \varepsilon) \tag{6}$$

where $I(x)$ is an indicator function which equals 1 if x is true and 0 otherwise, and ε is a tolerance threshold. The higher value of $rank(w_p, c)$, the lower accuracy a resulting word list has. Thus, our objective is to minimize the rank value of positive words, and formulated as follows.

$$\mathcal{O}_{(w_p,c)} = f(rank(w_p, c)) = \log_2(rank(w_p, c)). \tag{7}$$

Besides, according to our analysis in the previous section and the deduced optimization criterion given in Eq. 5, we intend to select the negative words that are likely to mess up our model since they have strong relations with positive words. Specifically, we opt to select the negative words that satisfy the following requirement:

$$v_c^\top v_{w_n} + \varepsilon > v_c^\top v_{w_p} \tag{8}$$

That is, we choose the negative words that violate the optimization criterion (see Eq. 5) in the last training iteration. Such kind of negative words can provide the most informative examples to strengthen our model in distinguishing very similar (positive and negative) words. Since the values of learned vectors v_c, v_{w_p}, v_{w_n} will be updated every iteration, our approach to sample negative words is a dynamic strategy.

Combing positive ranking and negative sampling together, we can obtain the following objective function to maximize the classification probability for positive words and minimize the probability difference for negative words at the same time.

$$\begin{aligned} \mathcal{J} = \; & \textstyle\sum_{(w,c)} \Big\{ \mathcal{O}_{(w_p,c)} \cdot \big\{ -\log(\sigma(v_c^\top v_{w_p})) \big\} \\ & + \textstyle\sum_{w_n \in N} \big\{ -\log(1 - \sigma(v_c^\top v_{w_n})) \big\} \Big\} \end{aligned} \tag{9}$$

when a positive word w_p is top ranked, the relevant rank value of $\mathcal{O}_{(w_p,c)}$ will be small, and the confidence to correctly

classify this example as positive will be also higher. Their multiplication will lead to a smaller value of the objective function.

Example. Figure 1 illustrates the process procedure (steps) of the CBOW and our OptRank models, taking as example the positive word 'France' mentioned in Table 1. Both models are trained in two steps. Specifically, CBOW will increase the relevance score of the positive word by the gradient values from 1.7 to 2.8. The score is still smaller than some other negative words, among which only the ranking scores of popular negative words (e.g., Cheap) are denoted by '*', leading to a better yet suboptimal ranking list after step 2. Although the ranking list is initially the same, the OptRank model increases the ranking score of the positive word to a larger extent with the help of item ranks at step 1. Then, OptRank adopts dynamic sampling to find an informative negative example (i.e., word cat) and decrease its ranking score. After that, the positive word will be ranked highest in this intuitive example.

3.2 Effective Learning Scheme

Next we present a learning scheme to effectively train our proposed model. We adopt the popular stochastic gradient descent (SGD) method to optimize Eq. 9. Specifically, for a given training word-context example (w_p, c), the gradient of our model parameter θ is given by:

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial \theta} &= \frac{\partial \mathcal{O}_{(w_p,c)} \cdot \{-\log(\sigma(v_c^\top v_{w_p}))\}}{\partial \theta} \\ &\quad + \sum_{w_n \in N} \sigma(v_c^\top v_{w_n}) \frac{\partial v_c^\top v_{w_n}}{\partial \theta} \qquad (10) \\ &\approx \mathcal{O}_{(w_p,c)} \cdot (\sigma(v_c^\top v_{w_p}) - 1) \frac{\partial v_c^\top v_{w_p}}{\partial \theta} \\ &\quad + \sum_{w_n \in N} \sigma(v_c^\top v_{w_n}) \frac{\partial v_c^\top v_{w_n}}{\partial \theta} \qquad (11)\end{aligned}$$

We are aware that Eq. 11 is not a standard gradient computation, because $\mathcal{O}_{(w_p,c)}$ is also related to model parameter θ, but we do not consider its derivatives. Similar idea and simplification are also adopted in [Weston *et al.*, 2010] and thus its usefulness has been verified. For each training example (w_p, c), we need to compute the ranking value of $rank(w_p, c)$, the exact value of which requires an exhaustive search in the whole word space. It is thus a very time-consuming step, and will become prohibitively expensive when being applied in a large-scale dataset.

To reduce the computational cost, we devise an approach to approximate the rank value by repeated sampling. Specifically, given a training example (w_p, c), we repeatedly sample a negative word from the corpus W until we obtain an expected word w_n that satisfy the requirement given by Eq. 8. That is, the ranking score of the negative word is greater than that of a positive word with tolerance value ε. Let k denote the number of sampling trials to retrieve a proper negative word. This number k follows a geometric distribution with parameter $p = \frac{rank(w_p,c)}{|corpus|} = \frac{rank(w_p,c)}{|W|}$. Then, the expectation of a geometrical distribution with parameter p is $\frac{1}{p}$, i.e., $k \approx \frac{1}{p} = \frac{|W|}{rank(w_p,c)}$. Thus, we can estimate the rank value by $rank(w_p, c) \approx \frac{|W|}{k}$. Similar idea has been used in [Yuan *et al.*, 2016] in a different problem setting. Therefore, we can rewrite Eq. 11 as follows:

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial \theta} &\approx f\Big(\frac{|W|}{k}\Big) \cdot \big(\sigma(v_c^\top v_{w_p}) - 1\big) \frac{\partial c_w^\top v_{w_p}}{\partial \theta} \\ &\quad + \sum_{w_n \in N} \sigma(v_c^\top v_{w_n}) \frac{\partial v_c^\top v_{w_n}}{\partial \theta} \qquad (12)\end{aligned}$$

Let $\theta = v_{w_p}$ or v_{w_n}, and then we obtain the following update rules:

$$v_{w_p} = v_{w_p} - \eta\Big(f\big(\frac{|W|}{k}\big)(\sigma(v_c^\top v_{w_p}) - 1)\Big)c_w \qquad (13)$$

$$v_{w_n} = v_{w_n} - \eta(\sigma(v_c^\top v_{w_n}))c_w \qquad (14)$$

3.3 Rank Normalization and Early Dropout

In the practical implementation, we notice that when the corpus scale reaches the level of hundreds of millions, the performance of our ranking model will decrease. The reason is that, in the early stage of training, we can easily find a negative word w_n that meet the requirement of Eq. 8, that is a small k value, leading to a very large $rank(w_p, c) \approx \frac{|W|}{k}$, often up to hundreds of millions. A consequence of large $f(rank(w_p, c))$ value is the problem of gradient explosion during the learning process. To solve this issue, we normalize the rank value as follows.

$$\overline{rank} = \frac{rank + \rho}{\varphi} \qquad (15)$$

where parameter φ is used to limit the upper range of the normalized rank, and thus its setting is often proportional to the corpus size. Note that if $rank < \varphi$, then $log_2(\frac{rank}{\varphi})$ will be smaller than 0. To avoid such a scenario, we use parameter ρ to adjust the overall rank value. Usually parameter ρ is set to $\varphi - 1$, or slightly greater than φ.

On the other hand, in the later stage of model training, it becomes difficult and takes longer to sample a proper negative word according to Eq. 8. In this case, we need to set up an exit mechanism to avoid resource occupation and reduce training time. To be specific, we will drop negative sampling when the number of sampling trials reaches a pre-defined threshold. Instead, we will adopt the negative word w_n with the maximal similarity (inner product) with the context v_c.

To sum up, the detailed pseudocode to train our OptRank model is illustrated in Algorithm 1. Specifically, we first randomly initialize variables v_m for all $w \in W$ by small values (line 1). The learning process will be repeatedly executed until reaching the maximal iterations (lines 3-21). In each iteration, we randomly select a positive training example (w_p, c) (line 3), and generate a vector to represent the contextual words v_c (line 6). In lines 8-10, we continue to sample a negative word unless it meets the demand of Eq. 8 or the number of sampling trials is greater than a threshold S. The rank value and corresponding objective is estimated in line 12. From lines 13-18, we adopt the gradient update rules to learn word embeddings for sampled negative words and the positive word. Lastly, lines 19-20 updates the vector representations of contextual words.

Algorithm 1: The OptRank learning algorithm

Randomly initialize variables $v_w, \forall w \in W$;
$t = 0$;
while $t <$ *MaxIteration* **do**
  Draw (w_p, c) uniformly;
  $e = 0, k = 0$;
  $v_c = \frac{1}{|c|} \sum_{p \in c} v_p$;
  for $u \in \{w_p\} \bigcup N$ **do**
    repeat
      Sample w_n;
      $k = k + 1$;
    until $v_{w_n}^\top v_c + \varepsilon > v_{w_p}^\top v_c$ *or* $k > S$;
    $\mathcal{O}_{(w,c)} = \log_2(\frac{\frac{|W|}{k} + \rho}{\varphi})$;
    if $u = w_p$ **then**
      $g = \eta\big(\mathcal{O}_{(w,c)}(\sigma(v_c^\top v_u) - 1)\big)$;
    else
      $g = \eta(\sigma(v_c^\top v_u))$;
    $e = e + g \cdot v_u$;
    $v_u = v_u - g \cdot v_c$;
  for $u \in c$ **do**
    $v_u = (v_u - e)/(|N| + 1)$;
  $t = t + 1$;

4 Evaluation

4.1 Experomrntal Setup

The training dataset used in our experiments is the Wikipedia 2017 articles (Wiki2017)[2], which contains around 2.3 billion words (14G). We sample a number of subsets from the corpus, the sizes of which are about 128M, 256M, 512M, 1G, 2G, respectively. After trained with a dataset, all comparison models are used to complete two widely-adopted tasks regarding word embeddings, namely word analogy and word similarity for the sake of performance evaluation.

Word Analogy Task

The word analogy task is to answer the questions in the form of "a is to b as c is to ?". Our testing set[3] consists of 19,544 such questions in two categories: semantic and syntactic. Specifically, the semantic questions are usually analogies regarding people name or locations. For instance, "Beijing is to China as Paris to ?". The syntactic questions are generally about verb tense or forms of adjectives, for example "Eat is to Eating as speak is to ?". Word embedding models need to predict the missing token for a given question, and thus are estimated if they can correctly retrieve the ground-truth word.

Word Similarity Task

Different from the word analogy task, this task does not require the exact match between the predicted token and the ground-truth word. Instead, it calculates the consine similarity between two words. The underlying assumption is that, it is acceptable for the performance of word embedding models that they can produce words similar enough, though it may not be an exact match. Six datasets are adopted as our testsets, namely WS353 [Finkelstein *et al.*, 2001], WS353R and WS353S [Agirre *et al.*, 2009], MTURK [Radinsky *et al.*, 2011], MEN [Bruni *et al.*, 2012] and Simlex999 [Hill *et al.*, 2015].

Comparison Methods

We implement and compare the following word embedding approaches with our OptRank model.

- CBOW-p [Mikolov *et al.*, 2013]: the original CBOW model with a popularity-based sampling strategy.
- CBOW-a [Chen *et al.*, 2017]: a CBOW variant which adaptively sample negative words by ranking scores.
- WordRank [Ji *et al.*, 2015]: a ranking model that puts more weights on positive words by rank values.
- OptRank: our ranking model with the optimization in both positive word ranking and negative word sampling.

Parameter Settings

For CBOW-p, CBOW-a and OptRank models, as suggested by [Mikolov *et al.*, 2013; Chen *et al.*, 2017], down-sampled rate is set to 0.001; the learning rate starts with $a = 0.025$ and changes by $a_t = a(1 - t/T)$, where T is the sample size and t is the iteration of current training examples. Besides, window size = 8, dimension = 300, and the size of negative sample is 15 in five subsets, and 2 in the whole Wiki2017 dataset, respectively. For the parameter *power* used in negative sampling, we find that $power = 0.75$ offers the best accuracy for CBOW-p and OptRank model, while $power = 0.005$ is suggested by [Chen *et al.*, 2017] and adopted for CBOW-a. Specially, the value of ε in OptRank should be adjust to the size of the corpus. We set ε as 0.5 in five subsets and 1.0 in Wiki2017(14G). For the WordRank model, we adopt the settings given by [Ji *et al.*, 2015]: logarithm as the objective function, initial value of scale parameter is $\alpha = 100$ and offset parameter $\beta = 99$. The dimension of word vectors is also set to 300.

4.2 Experimental Results

We mainly focus on the accuracy of two kinds of tasks mentioned above when comparing word embedding models.

Table 2 illustrates the accuracy of word embedding models along with the size variation of small training datasets. It is observed that our OptRank model consistently achieves much better results than the others across datasets and evaluation tasks. It can be explained by the fact that OptRank utilizes the associations between positive and negative words to weigh more on positive words and sample more informative negative words, while the other models take into account only one aspect either in word ranking or negative sampling. After training each model with the biggest dataset (14G Wiki2017) to near convergence, we proceed to compare the final accuracy on the two testing tasks and the results are illustrated in Figure 2 and Table 3, respectively. Some works [Le and Mikolov, 2014] contend that even using a small number of negative samples (e.g., 2 to 5) can achieve a respectable accuracy on large-scale datasets. Thus, we set negative samples to 2 (neg = 2) when using the whole Wiki2017 dataset.

[2] http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2
[3] https://github.com/tmikolov/word2vec

Corpus Size	Word Analogy				Word Similarity (average results on six testing datasets)			
	CBOW-p	CBOW-a	WordRank	OptRank	CBOW-p	CBOW-a	WordRank	OptRank
128M	0.364	0.404	0.415	0.437	0.622	0.618	0.633	0.637
256M	0.438	0.513	0.518	0.542	0.634	0.621	0.651	0.654
512M	0.543	0.632	0.642	0.658	0.643	0.637	0.657	0.675
1G	0.660	0.667	0.647	0.675	0.641	0.631	0.670	0.661
2G	0.691	0.712	0.685	0.718	0.647	0.646	0.665	0.672

Table 2: The best performance of each word embedding model in two testing tasks when the training datasets are relatively small

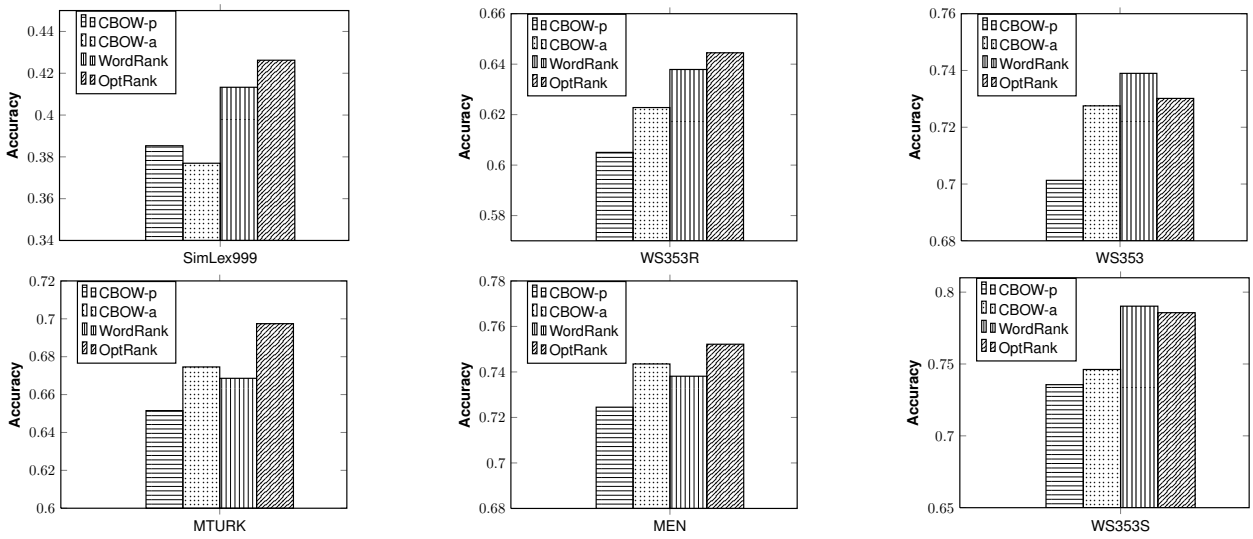

Figure 2: The best performance of each word embedding model (trained on 14G Wiki2017) for the task of word similarity

Model	Semantic	Syntactic	Overall
CBOW-p	0.782	0.678	0.725
CBOW-a	0.811	0.694	0.747
WordRank	0.775	0.687	0.722
OptRank	0.824	0.698	0.756

Table 3: The best perfomance of comparison models (trained on 14G Wiki2017) for the task of word analogy with neg = 2

For the task of word similarity, Figure 2 shows that the OptRank model consistently yields a much better performance than the CBOW-p and CBOW-a model, and beats WordRank on many datasets. Table 3 shows that OptRank is dominant on the word analogy task for all cases. We can observe that CBOW-a performs better than CBOW-p which is in turn better than WordRank. The reason is that WordRank only focuses on the ranks of positive words, but pays no attention to their differences relative to negative words.

5 Conclusion and Future Work

In this paper, we view word embedding as a ranking problem and then analyze the main disadvantage of CBOW model that it does not consider the relation between positive and negative words. This easily results in incorrect ranks of words, and produces suboptimal embeddings during training. Thus, we proposed a novel rank model which learns word representations not only by weighting positive words, but also by oversampling informative negative words. Other models typically only pay attention to one of them. Moreover, by using an effectively learning scheme, we reduce the computational cost of the OptRank, which makes it become a more practising model. These attributes significantly enable OptRank to achieve good performance even if the training datasets are limited.

Although our idea can be directly applied to the skip-gram model, the empirical study shows that the improvement is not as stable as CBOW. The reason is that there is only one target (positive) word in CBOW, but a set of positive words in skip-gram. Hence, in the future we intend to investigate how to handle the scenario with a set of target words. Meanwhile, we are also interested to compare our OptRank with a newly proposed embedding model Allvec [Xin *et al.*, 2018], which is learned by batch gradient descent with all negative examples instead of SGD with negative sampling.

Acknowledgments

This work was supported by the National Natural Science Foundation for Young Scientists of China under Grant No. (61702084, 61772125, 61702090) and the Fundamental Research Funds for the Central Universities under Grant No.N161704001.

References

[Agirre *et al.*, 2009] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, MariusPasca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[Bruni *et al.*, 2012] Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pages 136–145, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[Chen *et al.*, 2017] Long Chen, Fajie Yuan, Joemon M. Jose, and Weinan Zhang. Improving negative sampling for word representation using self-embedded features. *CoRR*, abs/1710.0980, 2017.

[Finkelstein *et al.*, 2001] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th International Conference on World Wide Web*, pages 406–414, New York, NY, USA, 2001. ACM.

[Hill *et al.*, 2015] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015.

[Ji *et al.*, 2015] Shihao Ji, Hyokun Yun, Pinar Yanardag, Shin Matsushima, and S. V. N. Vishwanathan. Wordrank: Learning word embeddings via robust ranking. *CoRR*, abs/1506.02761, 2015.

[Le and Mikolov, 2014] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 1188–1196, Bejing, China, 22–24 Jun 2014. PMLR.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[Mnih and Hinton, 2009] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc., 2009.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, volume 14, pages 1532 – 1543, 2014.

[Radinsky *et al.*, 2011] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, pages 337–346, New York, NY, USA, 2011. ACM.

[Weston *et al.*, 2010] Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.

[Xin *et al.*, 2018] xin Xin, Fajie Yuan, Xiangnan He, and Joemon Jose. Batch is not heavy: Learning word embeddings from all samples. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2018.

[Yuan *et al.*, 2016] Fajie Yuan, Guibing Guo, Joemon M. Jose, Long Chen, Haitao Yu, and Weinan Zhang. Lambdafm: Learning optimal ranking with factorization machines using lambda surrogates. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 227–236, New York, NY, USA, 2016. ACM.