Revisit Word Embeddings With Semantic Lexicons for Modeling Lexical Contrast

Jiawei Liu
University of Science and Technology of China, China
ustcljw@mail.ustc.edu.cn

Zhenyu Liu
China University of Political Science and Law Beijing, China
lzhy@cupl.edu.cn

Huanhuan Chen*
University of Science and Technology of China, China
hchen@ustc.edu.cn

***Abstract*—It is widely accepted that traditional word embedding models, which rely on distributional semantics hypothesis, are relatively limited for contrast meaning problem. Distributional semantics hypothesis indicates that words lying in similar contexts have similar representations in vector space. Nevertheless, synonyms and antonyms often locate in similar contexts, which means they appear close to each other in vector space. Hence, it is of great difficulty to distinguish antonyms from synonyms. To address this challenge, we propose an optimization model, named Lexicon-based Word Embedding Tuning (LWET) model. The goal of LWET is to incorporate reliable semantic lexicons to tune the distributions of pre-trained word embeddings in the vector space so as to improve their ability of distinguishing antonyms from synonyms. To speed up the training process of LWET, we propose two approximation algorithms, including positive sampling and quasi-hierarchical softmax. Compared with quasi-hierarchical softmax, positive sampling is faster, however, at the cost of worse performance. In experiments, LWET and other state-of-the-art models are tested on antonyms recognition, distinguishing antonyms from synonyms and word similarity. The results of the first two experiments show that LWET significantly improves the ability of word embeddings to detect antonyms, thus achieving the state-of-the-art performance. On word similarity, LWET gets slightly better performance than the state-of-the-art models. It means that LWET can remain and strengthen the semantic structure rather than destroy it when tuning word distributions in vector space. In general, compared with related work, LWET can not only achieve similar or even better performance, but also speed up the training process.**

I. Introduction

Word embedding, which is defined as learning word representations into vector space from collocations in large corpus, has been widely used in the area of Natural Language Processing (NLP). Traditional word embedding models contain Semantic Extraction using a Neural Network Architecture (SENNA) [1], the hierarchical log-bilinear model [2], Continuous Bag-of-Word Model (CBOW), Skip-gram [3] and Global Vectors [4]. Effectiveness of these models has been demonstrated by various applications, such as information retrieval [5], question answering [6] and text classification [7].

These models mentioned above are mainly based on the distributional semantics hypothesis, which indicates that words in similar context have similar representations in vector space. Nevertheless, it is reported that word embeddings trained by distributional semantics hypothesis are relatively limited in modeling lexical contrast [8]. Since a word's synonyms and its antonyms tend to share the same context, and they are likely to have similar representations. As a result, these word embeddings often lead to confusion between synonyms and antonyms. For example, in sentences "he likes the cat" and "he dislikes the cat", "likes" and "dislikes" sit in the same context. According to distributional semantics hypothesis, the two words will have similar representations. Hence, it is very difficult to distinguish them. Actually, modeling lexical contrast is a key point in NLP because it is strongly associated with other applications such as sentiment analysis [9] and text classification [7]. In order to overcome the deficiency of distributional semantics hypothesis in modeling lexical contrast, some effective models have been proposed [8], [10]–[13]. However, most of these models are time-consuming when applied to capture semantic information from a huge corpus.

Recent work shows that the performance of word embeddings can be significantly improved by incorporating other reliable linguistic resources [14]–[17] including semantic lexicons like PPDB [18], WordNet [19] and Rogets [20]. Motivated by the idea of using semantic lexicons, we propose an optimization model, called *Lexicon-based Word Embedding Tuning* (LWET) model, to improve the ability of word embeddings to detect contrast meanings. Based on the pre-trained word embeddings, we introduce a tuning process where reliable semantic lexicons are appended to tune the distributions of word embeddings in vector space so that the ability of word embeddings to distinguish antonyms from synonyms can be improved. For a target word, our strategy is to make synonyms locate in the nearest positions to the target word and antonyms far from the target word, while the irrelevant words act as a boundary locating between synonyms and antonyms. More details of LWET are illustrated in Fig.1.

In order to reduce the computational complexity of LWET, we propose 2 approximation algorithms: positive sampling and quasi-hierarchical softmax. The former is with higher speed and the latter can achieve better performance but at the cost of lower efficiency. We evaluate LWET and the state-of-the-art methods on three experiments: antonym recognition, distinguishing antonyms from synonyms and word similarity. The results of the first two experiments show that LWET can significantly improve the ability of word embeddings to detect antonyms. The performance on word similarity demonstrates

* indicates the corresponding author

978-1-5386-3120-1/17 $31.00 © 2017 IEEE
DOI 10.1109/ICBK.2017.35

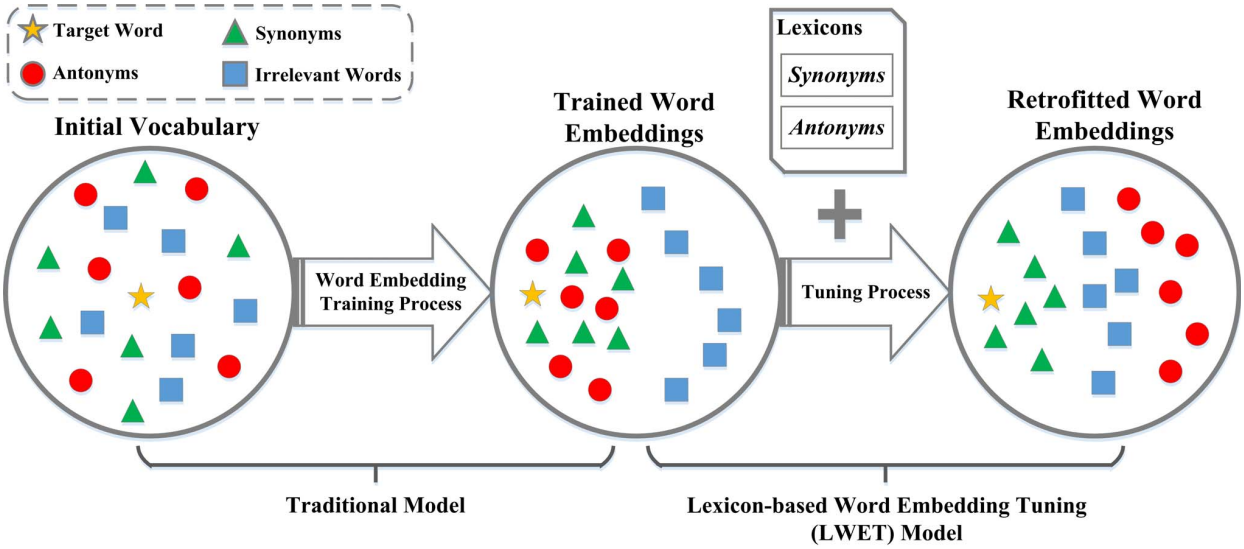

Fig. 1. The left part indicates the traditional model. After training process, synonyms and antonyms locate near with each other like the middle circle shows. The right part illustrates the basic idea of our model. In vector space, our model can make the antonyms far away from the target word so that synonyms and antonyms can be explicitly distinguished from each other.

that LWET can remain and strengthen the semantic structure rather than destroy it when tuning word distributions in vector space. For clarity, our contributions are summarized as follows:

- Actually, LWET is a linear model. Compared with related work, LWET is with lower computational complexity and can achieve similar or better performance.
- For solving LWET, we propose two approximation algorithms: positive sampling and quasi-hierarchical softmax. Both of the algorithms can speed up the training process. Positive sampling is quicker while quasi-hierarchical softmax can get better performance but at the cost of lower efficiency.
- LWET gets the state-of-the-art performance in the first two experiments and strengthens the original semantic structure at the same time.

The rest of the paper is arranged as follows. Section II introduces some related work. Lexicon-based Word Embedding Tuning model is proposed in Section III. In Section IV, we discuss the experiments used for performance comparison. Conclusion is drawn in Section V.

II. Related Work and Background

In dictionary, words "antonym", "contrasting" and "opposite" have different definitions. However these words sometimes can be used interchangeably. In this paper, they all refer to the same meaning when they are used.

Word Embedding Word embedding is to represent words with low dimensional vectors, which is a hot topic in NLP. Recently, some researchers are committed to find ways to improve the quality of word embeddings. Some of them modified existing models, and the others proposed some brand new models. In [21], global context information was added to input layer of their model to improve the quality of word embeddings. Recently, neural network-based models like CBOW and Skip-gram [3] have attracted a lot of attentions due to their effectiveness in dealing with large corpora. Pennington et al. presented a new global log-bilinear regression model named Glove that combines the advantages of global matrix factorization and local context window methods [4]. Liu et al. proposed a topic word embedding method, which learns word embeddings based on both words and their topics [7]. Sun et al. raised two novel distributional models combining not only syntagmatic but also paradigmatic relations [22]. Chen et al. utilized convolutional neural network to capture valuable information hidden in WordNet gloss and added it to training process of word embeddings [23].

Modeling Lexical Contrast Due to the deficiency of distributional semantics hypothesis-based word embeddings in modeling lexical contrast, Chen et al. proposed a contrasting word embedding framework and two effective models, which achieved the highest F-score of 92% on the Graduate Record Examination (GRE) "most contrasting word" questions [8]. In [12], Yih et al. utilized latent semantic analysis (LSA) to encode contrast meaning, which is called PILSA and achieved one of the highest F-score of 81% on the same dataset. Recently, Zhang et al. [24] proposed an efficient model, called Bayesian probabilistic tensor factorization. Not only the dimensional semantic information but also the other unsupervised lexicons information were considered in tensor factorization and this model achieved F-score of 82 % on the same dataset. Ono et al. [13] proposed a learning model which adds the thesauri information to Skip-gram with Negative Sampling (SGNS) and achieved F-score of 89% on the same dataset. However, these models mentioned above suffer from high computational complexity and need a lot of time to

conduct the training process. On the contrary, our model LWET is with much lower computational complexity and can train a large corpus in a short time.

III. Lexicon-based Word Embedding Tuning (LWET) Model

In this section, we are going to introduce Lexicon-based Word Embedding Tuning model (LWET). The objective of LWET is to incorporate semantic lexicons to improve the ability of word embeddings to detect contrast meanings. To speed up the training process of LWET, we proposed two approximation algorithms: positive sampling and quasi-hierarchical softmax. Both of the algorithms are introduced in the end of this section.

A. Model Description

For some word embedding models, which are related to model lexical contrast, the words in vocabulary are divided into only two parts: synonyms and antonyms. Basic idea of these models is to expand the distance between the target word and its antonyms. Although this method can help us easily distinguish antonyms from synonyms, the distance between antonyms and the irrelevant words is shortened as well, since the irrelevant words are always locate far away from the target word in initial vector space. Hence, it will be more difficult to distinguish antonyms from the irrelevant words.

In LWET, for a target word, the remaining words in vocabulary are divided into 3 categories: synonyms, antonyms and irrelevant words. In lexicons, irrelevant words are defined as the words, which are neither semantic-close nor semantic-opposite to the target word. As shown in Fig.1, the middle circle represents the structure of pre-trained word embeddings, where synonyms and antonyms of the target word are mixed together. Therefore, it is hard to distinguish them from each other. The right circle in Fig.1 illustrates the ideal result of LWET: synonyms become the nearest to the target word while antonyms stay in the furthest area, and the irrelevant words lie somewhere in between, which form a boundary to separate synonyms and antonyms.

Let $V = \{w_1, w_2, \cdots, w_n\}$ be the vocabulary, and n is the vocabulary size. The matrix $\widehat{\mathbf{Q}}$ denotes the pre-trained word embeddings, which are trained based on the models like Skip-gram, CBOW [3], Glove [4], etc. Each column $\widehat{q}_i$ ($\widehat{q}_i \in \mathbb{R}^d$) of $\widehat{\mathbf{Q}}$ is defined as the word vector of w_i, and d is the dimension of word vectors. The retrofitted vector representations are saved in a new collection matrix $\mathbf{Q} = \{q_1, \cdots, q_n\}$ ($q_i \in \mathbb{R}^d$). For a target word w_i, define $\mathbf{E}_S = \{(i,j) \mid w_i, w_j \in V$ and w_i, w_j *are synonyms* $\}$ as its synonymy relationship in vocabulary, $\mathbf{E}_A = \{(i,k) \mid w_i, w_k \in V$ and w_i, w_k *are antonyms* $\}$ as its antonymy relationship, and $\mathbf{E}_I = \{(i,l) \mid w_i, w_l \in V$ and w_i, w_l *are semantically unrelated* $\}$ as its irrelevant relationship.

After the tuning process, the retrofitted word vectors should be not only the closest to their counterparts and synonyms in $\widehat{\mathbf{Q}}$, but also far away from their antonyms. At the same time, the irrelevant words as a boundary locate between synonyms and antonyms. Ideally, the word distribution should meet the following conditions:

Condition 1: $\forall i \in [1,n], \mathfrak{D}(q_i, q_j) < \mathfrak{D}(q_i, q_l)$, $\forall (i,j) \in \mathbf{E}_S, \forall (i,l) \in \mathbf{E}_I$.

Condition 2: $\forall i \in [1,n], \mathfrak{D}(q_i, q_k) > \mathfrak{D}(q_i, q_l)$, $\forall (i,k) \in \mathbf{E}_A, \forall (i,l) \in \mathbf{E}_I$.

where $\mathfrak{D}(q_i, \widehat{q}_i)$, $\mathfrak{D}(q_i, q_j)$, $\mathfrak{D}(q_i, q_k)$, $\mathfrak{D}(q_i, q_l)$ represent the distance from word vector q_i to its counterpart in $\widehat{\mathbf{Q}}$, its synonym vector q_j, its antonym vector q_k and its irrelevant word vector q_l in $\mathbf{Q}$, respectively. From the pre-trained word embeddings, we observe that the distances between the target word w_i and its irrelevant words are usually the largest. In order to get an ideal result, the distances between w_i and its irrelevant words need to be shortened. Accordingly, during the tuning process, our optimal objective is to minimize the following cost function:

$$\begin{aligned}\Psi(\mathbf{Q}) = \sum_{i=0}^{n} \Big[\alpha \cdot \mathfrak{D}(q_i, \widehat{q}_i) + \beta \sum_{(i,j)\in\mathbf{E}_S} \frac{\mathfrak{D}(q_i, q_j)}{N_S^i} \\ - \gamma \sum_{(i,k)\in\mathbf{E}_A} \frac{\mathfrak{D}(q_i, q_k)}{N_A^i} + \delta \sum_{(i,l)\in\mathbf{E}_I} \frac{\mathfrak{D}(q_i, q_l)}{N_I^i} \Big]\end{aligned} \tag{1}$$

In the square brackets, the second component indicates the average distance between w_i and its synonyms, the third component denotes the average distance between q_i and its antonyms and the last component represents the average distance between w_i and its irrelevant words. N_S^i, N_A^i and N_I^i separately denote the number of synonyms, the number of antonyms and the number of irrelevant words of w_i. The parameters α, β, γ, δ control the relative strength of each component and need to be determined according to specific requirements. In this paper, we utilize Euclidean distance [14] to measure the relation between two words.

B. Learning

Although the synonyms and antonyms of a word are countable, the number of its irrelevant words is nearly equal to the vocabulary size n, which means the computational complexity of Eq. (1) is equal to $O(n^2)$ and can hardly be solved in finite time. To reduce the computational complexity, we propose two approximation algorithms: positive sampling and quasi-hierarchical softmax.

1) Positive Sampling: This method is motivated by the strategy "negative sampling" [25] where non-context words are regarded as "negative" samples. Here, the distance $\mathfrak{D}(q_i, q_l), (i,l) \in \mathbf{E}_I$ also need to be shortened to some extent, which is similar to the operations of q_i and its synonymy vectors. Therefore, the irrelevant words are regarded to be "positive" samples. For word w_i, we assume that its irrelevant words are uniformly distributed in vector space and randomly sample m_i words from its original irrelevant word set as its new irrelevant word set, as shown in the following equation:

$$\begin{aligned}\mathbf{C}(w_i) = sample(i, m_i) \subseteq \mathbf{E}_I,\ i \in [1,n], \\ m_i = \max\{N_A^i, N_S^i\},\end{aligned} \tag{2}$$

where $sample(i, m_i)$ means randomly sampling m_i irrelevant connections from E_I. If word w_i has no antonym or synonym,

Algorithm 1 LWET with Positive Sampling

Input: Initial word embeddings $\widehat{Q}$, Learning rate $step$, Threshold λ, Iterating times t
Output: Word embeddings which have been improved Q
Initialize: $\alpha, \beta, \gamma, \delta$, $\Psi_{old} = 0$, $\Psi_{new} = 0$, $Q = \widehat{Q}$, $index = 0$
while TRUE **do**
  $index = index + 1$
  $\Psi_{old} = \Psi_{new}$
  $\Psi_{new} = 0$
  for w_i in V **do**
    get w_i's synonyms from lexicons, S_{w_i}
    get w_i's antonyms from lexicons, A_{w_i}
    sample m irrelevant words and get $C(w_i)$
    update Ψ_{new} according to cost function.
    update the first derivative of $\Psi(Q)$ with respect to w_i
    update w_i's vector according to the first derivative
  end for
  if $\Psi_{new} - \Psi_{old} < \lambda$ or $index > t$ **then**
    BREAK
  end if
end while
return Q

Algorithm 2 LWET with Quasi-hierarchical Softmax

Input: Initial word embeddings $\widehat{Q}$, Learning rate $step$, Threshold λ, Iterating times t
Output: Word embeddings which have been improved Q
Initialize: $\alpha, \beta, \gamma, \delta$, $\Psi_{old} = 0$, $\Psi_{new} = 0$, $Q = \widehat{Q}$, $index = 0$
while TRUE **do**
  $index = index + 1$
  $\Psi_{old} = \Psi_{new}$
  $\Psi_{new} = 0$
  for w_i in V **do**
    get w_i's synonyms from lexicons, S_{w_i}
    get w_i's antonyms from lexicons, A_{w_i}
    get irrelevant words I_{w_i}, $I_{w_i} = V - S_{w_i} - A_{w_i}$
    divide I_{w_i} into $\left\lceil \sqrt{N_I^i} \right\rceil$ parts and calculate the central points q_t of each part
    update Ψ_{new} according to cost function.
    update the first derivative of $\Psi(Q)$ with respect to w_i
    update w_i's vector according to the first derivative
  end for
  if $\Psi_{new} - \Psi_{old} < \lambda$ or $index > t$ **then**
    BREAK
  end if
end while
return Q

m_i is set as 0, otherwise m_i is equal to the max value of N_A^i and N_S^i. Thus, the fourth part of Eq. (1) can be approximated as ($m_i \neq 0$):

$$\delta \sum_{(i,l)\in \mathbf{E}_I} \frac{\mathfrak{D}(q_i, q_l)}{N_I^i} \cong \delta \sum_{(i,l)\in \mathbf{C}(w_i)} \frac{\mathfrak{D}(q_i, q_l)}{m_i}, \tag{3}$$

Hence, the computational complexity of modified Eq. (1) becomes $O(nm)$, where $m = \frac{1}{n}\sum_{i=1}^{n} m_i$.

2) Quasi-hierarchical Softmax: This method is inspired by hierarchical softmax [25]. For word w_i, we equally separate its N_I^i irrelevant words into T groups and each group contains nearly $\left\lceil \sqrt{N_I^i} \right\rceil$ words. In each group, the central point of all words q_t ($t \in [1, T], T = \left\lceil \sqrt{N_I^i} \right\rceil$) is figured out as the representative for the group. Therefore, we can utilize T derived vectors to approximately calculate the fourth part of Eq. (1):

$$\delta \sum_{(i,l)\in \mathbf{E}_I} \frac{\mathfrak{D}(q_i, q_l)}{N_I^i} \cong \delta \sum_{t=1}^{T} \frac{\mathfrak{D}(q_i, q_t)}{T}. \tag{4}$$

Thus, the computational complexity of Eq. (1) can be reduced to $O(n\sqrt{n})$. With the modified cost function, we can employ stochastic gradient decent to tune each word embedding q_i ($i \in [1, n]$) iteratively. Hyperparameters α, β, γ, δ are chosen based on grid searching.

The algorithm descriptions of positive sampling and quasi-hierarchical softmax are shown in Algorithm 1 and Algorithm 2, respectively.

IV. Experiments

In this section, we are about to introduce the experiments used to evaluate the performance of LWET. Among all experiments, antonym recognition and distinguishing antonyms from synonyms are conducted for testing the ability of word embeddings to do contrast meanings detection. Word similarity is utilized to show whether LWET will destroy the semantic structures in original word vector space. In the end, we discuss the effect of hyperparameters on the quality of word embeddings. For clarity, some experimental settings are given firstly.

A. Experimental Settings

In this paper, we compare LWET with the baselines: CBOW, Skip-gram [3] and Glove [4], which are all based on distributional semantics hypothesis. All baselines are famous for their efficiency and effectiveness. Glove is trained based on the source code[1] and the rest baselines are trained by word2vec[2] using negative sampling algorithm. The corpus utilized in this paper origins from the 2013 ACL Workshop on Machine Translation[3], which is used in [26]. It contains nearly 500 millions of tokens and 600 thousands of vocabularies. The lexicons we adopt are provided by Chen et al. [8], which contain 92339 word types, 520734 antonym pairs and 646433 synonym pairs. The dimension of word embeddings is set as 300, which is reported to work well in [3]. The number of

[1]http://nlp.stanford.edu/projects/glove/
[2]https://github.com/dav/word2vec
[3]http://www.statmt.org/wmt13/translation-task.html

negative samples is set as 20 according to the empirical rule mentioned in [25]. Context window size is set as 5. Other parameters are equal to the default settings in the source codes. Based on grid searching, in LWET, we set $\alpha = 1$, $\beta = 2$, $\gamma = 3$ and $\delta = 4$. We observe that the retrofitted word embeddings perform better than all the baselines no matter which approximation algorithm is selected.

Compared with quasi-hierarchical softmax, the tuning process can work faster with positive sampling but at the cost of worse performance. For fair comparison, we report our best performance which is based on quasi-hierarchical softmax algorithm.

TABLE I
THE RESULTS OF ANTONYM RECOGNITION. "+" MEANS THE RESULTS OF LWET IS BASED ON THE ADJACENTLY LEFT-SIDE TRAINING PROCESS. NUMBERS IN BOLD MEAN THE BEST ANSWERS.

	CBOW	+LWET	SG	+LWET	Glove	+LWET
Dev.	0.16	0.59	0.10	**0.61**	0.15	0.56
Test	0.13	0.63	0.12	**0.65**	0.13	0.60
Adj.	0.32	**0.86**	0.42	0.86	0.41	0.86
Verb	0.39	0.91	0.29	**0.94**	0.31	0.88
Noun	0.36	0.81	0.43	**0.82**	0.34	0.81
Adv.	0.29	**0.92**	0.41	0.92	0.37	0.92

B. Antonym Recognition

The experiment is conducted to test the ability of word embeddings to distinguish antonyms. We utilize the "closest-to-opposite" dataset, which is widely used in previous work [8], [12]. The dataset includes 2 parts: development set and test set. The former contains 162 questions and the latter contains 790 questions [11]. Moreover, we also use another "closest-to-opposite" question dataset which is created for WordNet opposites. This dataset consists of 4 parts including adjectives (551 questions), adverbs (165 questions), nouns (330 questions) and verbs (226 questions). All the datasets are kindly released at Mohammad's homepage[4]. Each question in the datasets has three components including target word, five candidates and the correct answer. This task is to find the antonym of target word from the candidates. In the vector space of retrofitted word embeddings, the candidate with the largest Euclidean distance to the target word is chosen as the answer. For question "$a :: c_1, c_2, c_3, c_4, c_5 :: ans$", the target of this task can be organized as following equation:

$$\widehat{ans} = arg\max_{c_i} \ Euc(a, c_i), i \in [1, 5] \tag{5}$$

where $Euc(a, c_i)$ denotes the Euclidean distance between a and c_i. If $\widehat{ans} = ans$, the question is identified to be correctly answered. For pre-trained word embeddings, the word whose distance to the target word is the smallest is viewed as answer. We apply F-score to evaluate the performance of word embeddings. The results are reported in Table I.

From Table I, we find that the baselines achieve poor performance on all datasets, which validates the drawback of distributional semantics hypothesis when modeling lexical contrast. On the contrary, LWET significantly improves the performance, which illustrates the effectiveness of our model.

C. Distinguishing antonyms from synonyms

This experiment is conducted to evaluate the ability of word embeddings to distinguish antonyms from synonyms. To some extent, this experiment is more difficult to the first one. We utilize the dataset mentioned in [27], which contains 3 parts: adjectives (300 antonymous pairs and 300 synonymous pairs), nouns (350 antonymous pairs and 350 synonymous pairs) and verbs (400 antonymous pairs and 400 synonymous pairs). The accuracy is applied to evaluate the performance. To calculate the accuracy, we firstly sort the synonymous and antonymous word pairs according to their Euclidean distances. If a word pair belongs to the first half, it is viewed as synonymous pair. An antonymous pair is confirmed if it is in the last half. The results are reported in Table III.

Compared with the results described in [27], LWET gets better performance. Therefore, LWET gets the state-of-the-art result on this task, which consistently demonstrates the effectiveness of LWET.

TABLE II
THE RESULTS OF WORD SIMILARITY. "+" MEANS THE RESULTS OF LWET IS BASED ON THE ADJACENTLY LEFT-SIDE TRAINING PROCESS. RESULTS ARE REPORTED BASED ON SPEARMAN'S RANK.

	CBOW	+LWET	SG	+LWET	Glove	+LWET
RG-65	0.63	0.64	0.68	**0.70**	0.60	0.62
WS-353	0.60	0.61	0.65	**0.66**	0.49	0.50
Men-3k	0.68	0.68	0.72	**0.73**	0.61	0.61
SCWS	0.63	**0.64**	0.63	0.64	0.62	0.62
RW	0.44	0.45	0.46	**0.46**	0.43	0.44
MTurk287	0.65	**0.67**	0.66	0.67	0.65	0.65

D. Effect of Modeling Contrast on Semantic Similarity

According to our strategy, LWET can adjust the positions of words in vector space, which probably does harm to the original semantic structure. This experiment, which is also called word similarity, is conducted to test whether LWET has a negative effect on the semantic structure in initial vector space. We choose 6 gold standard datasets: Wordsim-353 [28], RG-65 [29], Men-3k [30], Rare-Word [31], SCWS [21] and MTurk287 [32]. Each dataset consists of 2 parts: word pairs and human score. After obtaining the Euclidean distance of each word pair, we evaluate the correlation between the distance and human score. We apply Spearman's rank correlation coefficient to quantify the results , which are shown in Table II. It is obvious that LWET maintains and slightly enhances the semantic information rather than destroys it.

[4]http://www.saifmohammad.com/WebPages/ResearchInterests.html

TABLE III

RESULTS OF DISTINGUISHING ANTONYMS FROM SYNONYMS."S" DENOTES SYNONYMOUS PAIRS AND "A" DENOTES ANTONYMOUS PAIRS. RESULTS ARE REPORTED BASED ON ACCURACY. "+" MEANS THE RESULTS OF LWET IS BASED ON THE ADJACENTLY LEFT-SIDE TRAINING PROCESS. NUMBERS IN BOLD MEAN THE BEST ANSWERS.

		CBOW	+LWET	SG	+LWET	Glove	+LWET
Verb	S	0.53	0.68	0.58	**0.69**	0.55	0.66
	A	0.53	0.68	0.58	**0.69**	0.55	0.66
Noun	S	0.43	0.55	0.51	0.55	0.53	**0.63**
	A	0.43	0.55	0.51	0.55	0.53	**0.63**
Adj.	S	0.54	**0.78**	0.54	0.77	0.56	0.77
	A	0.54	**0.78**	0.54	0.77	0.56	0.77

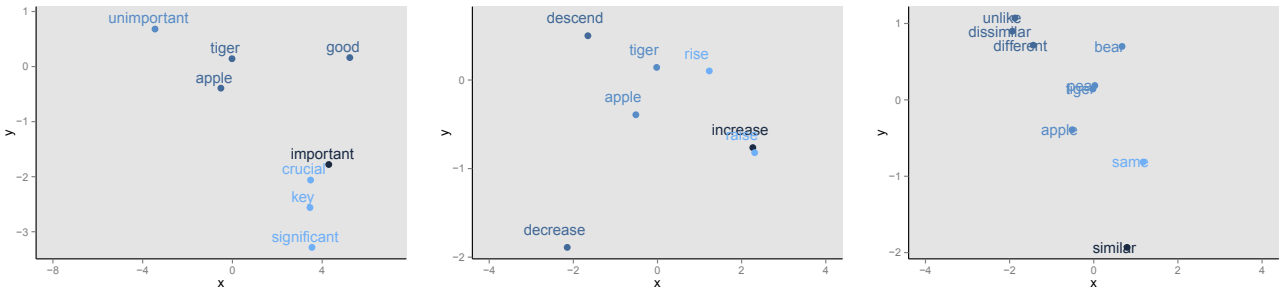

Fig. 2. Semantic structure of LWET's word embeddings. Words "important", "increase" and "similar" are the target words. Obviously, synonyms are the nearest to the target word. Antonyms are far away from the target word. Irrelevant words like "tiger", "bear", "pear" and "apple" are always located in the area between synonyms and antonyms.

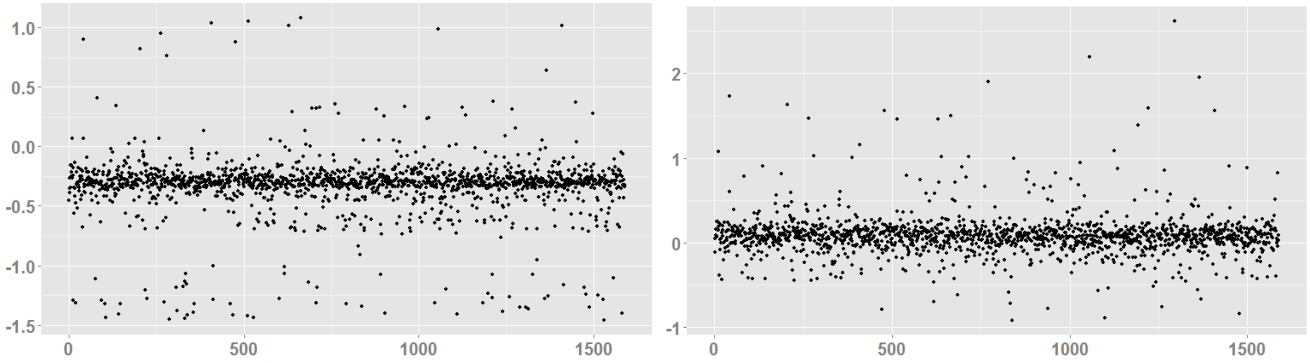

Fig. 3. In this figure, we illustrate the difference between the average antonymous distance and the average irrelevant distance. The left figure illustrates the result of pre-trained word embeddings. Most of the points are below zero, which indicates that the irrelevant words are always in the farthest position. The right figure is ours. After tuning process, antonyms locate in the farthest position.

E. Semantic Structure In Vector Space of LWET

In this section, we exploit *Principal Component Analysis* (PCA) to reduce the dimension of LWET's word embeddings from 200 to 2. The pre-trained word embeddings used in LWET is based on Skip-gram. We randomly select some words and their antonyms, synonyms and irrelevant words from the vocabulary and illustrate them in Fig.2 by using some visualization tools. From the pictures, we can easily find some meaningful semantic structures. Firstly, synonyms are always near to the target word. Secondly, the distance between the target word and its antonyms is always the largest. Lastly, irrelevant words are like a boundary between synonyms and antonyms, which can help us easily distinguish antonyms from synonyms. These exciting semantic structures are corresponding to our expectation and consistent with the objective of LWET.

F. Parameter Analysis

In order to validate LWET, we extract the target words, which have synonyms and antonyms in lexicons, from all datasets of antonym recognition experiment. Then, we calculate the average distances between each word and its synonyms, antonyms and irrelevant words, respectively. We find

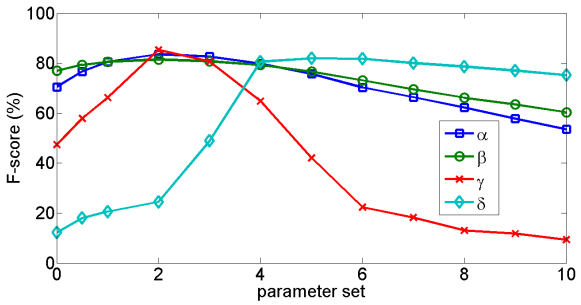

Fig. 4. Parameter Analysis of LWET based on antonym recognition. The performance is sensitive to γ and δ.

the average distance between target word and its synonyms are always the smallest. We use the average distance between a word and its antonyms minus the average distance between it and its irrelevant words. The difference is illustrated in Fig.3. The x-axis is the order of target words. The y-axis describes this difference. The left figure illustrates the original word embeddings and the right figure is ours. From the left figure, we observe that most of the points are below zero, which means that the irrelevant words are always in the farthest distance. After our revision, we observe that most of the points are larger than zero in the right figure, which means the antonyms are always in the farthest distance.

We also do parametric sensitivity analysis of LWET. Firstly, all datasets of antonyms recognition are combined. Then, we detect the F-score of antonyms recognition on the new dataset when we change the parameters. Each parameter varies from 0 to 10, and we change only one parameter each time. Result is illustrated in Fig.4. As observed, the result is not very sensitive to α and β. When γ changes from 2 to 6 and δ changes from 2 to 4, however, the result suffers great changes. In other word, the result mainly depends on γ and δ. According to our observation, parameter α and β should be chosen from 1 to 3, γ should be chosen from 2 to 3 and δ should be chosen from 4 to 6.

V. Conclusion

To overcome the deficiency of distributional semantics hypothesis when modeling lexical contrast, in this paper, we proposed a novel model named Lexicon-based Word Embedding Tuning model (LWET). The goal of LWET is to utilize semantic lexicons to adjust the distributions of words in vector space so that the ability of word embeddings to distinguish antonyms and synonyms can be improved. For a target word, our strategy is to make the synonyms become the nearest to it while antonyms stay in the furthest area, and the irrelevant words as a boundary lie somewhere in between. To solve LWET, we propose 2 approximation algorithms: positive sampling and quasi-hierarchical softmax. Positive sampling is with faster training speed but at the cost of worse performance than quasi-hierarchical softmax. We test LWET together with the baselines on antonym recognition, distinguishing antonyms from synonyms and word similarity. The first two experiments indicate that LWET can significantly improve the ability of word embeddings to detect antonyms. The last experiment shows that LWET can remain and enhance the semantic similarity in vector space rather than destroy these valuable semantic information. In general, based on semantic lexicons, LWET can significantly improve the ability of word embeddings to distinguish synonyms and antonyms. The word embeddings used in this paper are trained based on a medium-size corpus. In future work, we will retrain all models on a huge corpus and verify LWET again.

VI. Acknowledgement

This work is supported by the National Key Research and Development Program of China with grant number 2016YFB1000905, the National Science Foundation of China with grant numbers 91546116, 61673363 and the Fundamental Research Funds for the Central Universities with grant number WK2150110001.

References

[1] R. Collobert and J. Weston, "Fast semantic extraction using a novel neural network architecture," in *Annual Meeting Association for Computational Linguistics*, vol. 45, no. 1, 2007, p. 560–567.

[2] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," in *Advances in Neural Information Processing Systems*, 2009, pp. 1081–1088.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[4] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *Conference on Empirical Methods in Natural Language Processing*, vol. 14, 2014, pp. 1532–1543.

[5] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1, no. 1.

[6] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton, "Quantitative evaluation of passage retrieval algorithms for question answering," in *SIGIR 2003: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, 2003, pp. 41–47.

[7] Y. Liu, Z. Liu, T.-S. Chua, and M. Sun, "Topical word embeddings," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015, pp. 2418–2424.

[8] Z. Chen, W. Lin, Q. Chen, X. Chen, S. Wei, H. Jiang, and X. Zhu, "Revisiting word embedding for contrasting meaning," in *Annual Meeting Association for Computational Linguistics*, 2015, pp. 106-115.

[9] B. Pang, L. Lee *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[10] S. Mohammad, B. Dorr, and G. Hirst, "Computing word-pair antonymy," in *the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2008, pp. 982–991.

[11] S. M. Mohammad, B. J. Dorr, G. Hirst, and P. D. Turney, "Computing lexical contrast," *Computational Linguistics*, vol. 39, no. 3, pp. 555–590, 2013.

[12] W.-t. Yih, G. Zweig, and J. C. Platt, "Polarity inducing latent semantic analysis," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 2012, pp. 1212–1222.

[13] M. Ono, M. Miwa, and Y. Sasaki, "Word embedding-based antonym detection using thesauri and distributional information," in *Proceedings of the North American Chapter of the Association for Computational Linguistics*, 2015.

[14] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," *arXiv preprint arXiv:1411.4166*, 2014.

[15] M. Yu and M. Dredze, "Improving lexical embeddings with semantic knowledge," in *Association for Computational Linguistics (ACL)*, 2014, pp. 545–550.

[16] J. Bian, B. Gao, and T.-Y. Liu, "Knowledge-powered deep learning for word embedding," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 132–148.

[17] C. Xu, Y. Bai, J. Bian, B. Gao, G. Wang, X. Liu, and T.-Y. Liu, "Rc-net: A general framework for incorporating knowledge into word representations," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 1219–1228.

[18] J. Ganitkevitch, B. Van Durme, and C. Callison-Burch, "Ppdb: The paraphrase database." in *HLT-NAACL*, 2013, pp. 758–764.

[19] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[20] B. A. Kipfer, *Roget's 21st century thesaurus in dictionary form: the essential reference for home, school, or office*. Laurel, 1993.

[21] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 2012, pp. 873–882.

[22] F. Sun, J. Guo, Y. Lan, J. Xu, and X. Cheng, "Learning word representations by jointly modeling syntagmatic and paradigmatic relations," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2015, pp. 136–145.

[23] T. Chen, R. Xu, Y. He, and X. Wang, "Improving distributed representation of word sense via wordnet gloss composition and context clustering." in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2015, pp. 15–20 .

[24] J. Zhang, J. Salwen, M. R. Glass, and A. M. Gliozzo, "Word semantic representations using bayesian probabilistic tensor factorization." in *the Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1522–1531.

[25] T. Mikolov and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111-3119.

[26] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *arXiv preprint arXiv:1508.06615*, 2015.

[27] K. A. Nguyen, S. S. I. Walde, and N. T. Vu, "Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction," in *Annual Meeting Association for Computational Linguistics*, 2016, pp. 454–459 .

[28] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 406–414.

[29] H. Rubenstein and J. B. Goodenough, "Contextual correlates of synonymy," *Communications of the ACM*, vol. 8, no. 10, pp. 627–633, 1965.

[30] E. Bruni, N.-K. Tran, and M. Baroni, "Multimodal distributional semantics." *Journal of Artificial Intelligience Research*, vol. 49, no. 1, pp. 1–47, 2014.

[31] T. Luong, R. Socher, and C. D. Manning, "Better word representations with recursive neural networks for morphology." in *The SIGNLL Conference on Computational Natural Language Learning*, 2013, pp. 104–113.

[32] G. Halawi, G. Dror, E. Gabrilovich, and Y. Koren, "Large-scale learning of word relatedness with constraints," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 1406–1414.