Cogn Comput (2017) 9:843–851
DOI 10.1007/s12559-017-9492-2

Learning Word Representations for Sentiment Analysis

Yang Li[1] · Quan Pan[1] · Tao Yang[1] · Suhang Wang[2] · Jiliang Tang[3] · Erik Cambria[4]

Received: 29 April 2017 / Accepted: 18 July 2017 / Published online: 17 August 2017

Abstract Word embedding has been proven to be a useful model for various natural language processing tasks. Traditional word embedding methods merely take into account word distributions independently from any specific tasks. Hence, the resulting representations could be sub-optimal for a given task. In the context of sentiment analysis, there are various types of prior knowledge available, e.g., sentiment labels of documents from available datasets or polarity values of words from sentiment lexicons. We incorporate such prior sentiment information at both word level and document level in order to investigate the influence each word has on the sentiment label of both target word and context words. By evaluating the performance of sentiment analysis in each category, we find the best way of incorporating prior sentiment information. Experimental results on real-world datasets demonstrate that the word representations learnt by DLJT2 can significantly improve the sentiment analysis performance. We prove that incorporating prior sentiment knowledge into the embedding process has the potential to learn better representations for sentiment analysis.

✉ Erik Cambria
cambria@ntu.edu.sg

Yang Li
liyangnpu@mail.nwpu.edu.cn

Quan Pan
quanpan@nwpu.edu.cn

Tao Yang
yangtao107@nwpu.edu.cn

Suhang Wang
swang187@asu.edu

Jiliang Tang
tangjili@msu.edu

[1] School of Automation, NorthWestern Polytechnical University, Xián, ShanXi, 710072, People's Republic of China

[2] Department of Computer Science and Engineering, Arizona State University, Tempe, AZ, 85281, USA

[3] Computer Science and Engineering, Michigan State University, East Lansing, MI, 48824, USA

[4] School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Ave, Singapore, 639798, Singapore

Introduction

Word embedding is a popular method for natural language processing (NLP) that aims to learn low-dimensional vector representations of words from documents. Due to its ability to capture syntactic and semantic word relationships, word embedding algorithms such as Skip-gram, CBOW [17, 18] and GloVe [23] have been proven to facilitate various NLP tasks, such as word analogy [18], parsing [1], POS tagging [13], aspect extraction [25], temporal tagging [35], personality recognition [16], and multimodal fusion [27]. The assumption behind these word embedding approaches is the distributional hypothesis that "you shall know a word by the company it keeps" [8]. By leveraging on statistical information such as word co-occurrence frequencies, word embedding approaches can learn distributional vector representations that capture semantic meanings of words.

The majority of existing word embedding algorithms merely take into account statistical information from documents [17, 18, 23]. The representations learnt by such algorithms are very general and can be applied to various tasks. However, they are trained completely independently

from any specific task; thus, they may not be optimal for a given NLP task, especially when prior knowledge or auxiliary information about such a task is available. Recent advances in document representation show that incorporating auxiliary information such as document labels or links for learning document representations can lead to better document classification performance [31, 32]. Thus, training word embedding for specific tasks by incorporating prior knowledge should always be done when such knowledge is available.

In sentiment analysis, there are various types of prior knowledge available. For example, many datasets provide sentiment labels for documents, which can be used to form the word-sentiment distribution, i.e., the empirical probability each word has to appear in documents with a particular sentiment. External sources can also provide prior knowledge. For example, the sentiment polarities of words can be obtained via some public sentiment lexicons such as the multi-perspective question answering (MPQA) corpus [34], NRC [20], and SenticNet [4], which have been demonstrated to be useful for sentiment analysis. Prior knowledge carries complementary information that is not available in word co-occurrence and, hence, may greatly enhance word embeddings for sentiment analysis. For example, the words "like" and "dislike" can appear in the same or similar context such as *I love reading books* and *I dislike reading books*. By merely looking at word co-occurrences, we would learn similar vector representations of "like" and "dislike" as these have similar lexical behavior. From a sentiment point of view, however, such vector representation should be very different as they convey opposite polarity. Hence, by incorporating prior sentiment knowledge about these two words, we can build more sentiment-aware word embeddings and, hence, learn better distributional representations for sentiment analysis.

In this paper, we study the problem of learning word embeddings for sentiment analysis by exploiting prior knowledge during the embedding learning process. In doing this, we faced two main challenges: (1) how to mathematically represent prior knowledge; and (2) how to efficiently incorporate the prior knowledge into word embedding learning process. In an attempt to solve such challenges, we propose novel models that utilize word- and document-level sentiment prior knowledge. The main contributions of the paper are listed as follows:

- Providing a principled way to represent sentiment prior knowledge from both document and word level;
- Proposing a novel framework that incorporates different levels of prior knowledge into the word embedding learning process for better sentiment analysis; and
- Conducting extensive experiments to demonstrate the effectiveness of the proposed framework and investigating which types of prior knowledge work better.

The rest of the paper is organized as follows: "Related Works" reviews related works in the context of sentiment analysis; "The Proposed Framework" introduces the proposed model in detail and proposes a complexity analysis; in "Experimental Results", we conduct experiments to demonstrate the effectiveness of the proposed models; finally, "Conclusion" concludes the paper and proposes future work.

Related Works

Word embedding is a model that aims to learn low-dimensional continuously-valued vector representations of words and has attracted increasing attention [2, 6, 11, 17, 19, 23]. The learnt embedding is able to capture semantic and syntactic relationships and thus facilitate many NLP tasks such as word analogy [18], parsing [1], sentiment analysis [12], and machine translation [37]. The majority of existing word embedding algorithms such as Skip-gram, CBOW[17] and GloVe [23] follow the essential idea of distributional hypothesis [8], which states that words occurring in the same contexts tend to have similar meanings. Based on this hypothesis, Skip-gram optimizes word representations by finding a center word, which is good at predicting its neighboring words, and CBOW is good at predicting the center word given the neighboring words. GloVe investigates the ratios of word co-occurrence probabilities for learning word embedding using weighted least square. These algorithms merely utilize statistical information such as word co-occurrence from documents in an unsupervised setting. However, for a given specific task, there is also auxiliary information available, which has been proven to be helpful for learning task-specific word embeddings that can improve the performance of such a task [14, 36]. For example, TWE proposed in [14] exploits the topic information derived from latent Dirichlet allocation (LDA) for learning word embeddings that capture topical information and outperform CBOW on document classification.

Sentiment analysis [3] is a branch of affective computing [24] that requires tackling many NLP sub-tasks, including subjectivity detection [5], concept extraction [28], named entity recognition [15], and sarcasm detection [26]. Many sentiment analysis algorithms have exploited prior knowledge to improve the classification performance [9, 10, 33]. For example, emotional signals such as "lol" and emoticons are utilized for sentiment analysis in [10] and [9], respectively. USEA proposed in [33] uses the publicly available sentiment lexicon MPQA as the indication of word sentiment for unsupervised image sentiment analysis. Although it is of great potential to learn better word embeddings for sentiment analysis by incorporating prior knowledge, the work on this is rather limited. SE-HyRank in [30] utilizes

lexical-level sentiment supervision from Urban Dictionary and proposed a neural network based model for learning word embeddings. The proposed framework is different from SE-HyRank as (1): we investigate various types of prior knowledge, i.e., sentiment distribution and ratios of polarity from labels and sentiment weights from sentiment lexicons; and (2) we propose a novel framework based on GloVe, which combines the advantage of global matrix factorization, local context window methods and sentiment prior knowledge; and (3) we investigate which way of incorporating prior knowledge is better.

The Proposed Framework

Throughout the paper, P stands for the probability and W_i denotes the i-th word in the vocabulary. X_{ij} is the co-occurrence number of W_i and W_j in a context window of size p. Next, we will introduce the details of the proposed framework.

A Basic Model – GloVe

GloVe is one of the most popular word embedding algorithms [23]. The intuition of GloVe is that good word vectors can be learnt from the ratios of co-occurrence probabilities. In particuar, considering three words W_i, W_j and W_k, let P_{ik} denote the probability of W_k being the context word of W_i, i.e., W_k is a neighboring word of W_i with W_i at the center of a context window. Then, if $\frac{P_{ik}}{P_{jk}}$ is close to 1, the vector representations of W_i and W_j should be similar to the vector representation of W_k. On the other hand, if $\frac{P_{ik}}{P_{jk}}$ is large, then the vector representations of W_i and W_j should be far away from W_k. To reflect the information captured in the ratio of co-occurrence probabilities, GloVe formulates the intuition as follows

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}} \tag{1}$$

where w_i and w_j are vector representations of the target words W_i and W_j while $\tilde{w}_k$ is the vector representation of the context word W_k. F is a function that we can reflect the information contained in $\frac{P_{ik}}{P_{jk}}$. However, considering two target words, say W_i ="like" and W_j="dislike", obviously, we would have $\frac{P_{ik}}{P_{jk}} \approx 1$ for these two words as the context of these two words is very similar. Thus, by using Eq. 1, we would learn similar vector representations for "like" and "dislike," which are not correct for sentiment analysis as they have opposite sentiment polarities. Thus, we should take the prior sentiment information of words into consideration for learning more meaningful word embeddings.

Incorporating Prior Sentiment Knowledge

There are many different types of prior sentiment information available, which can be generally classified into two categories, i.e., *document-level sentiment* and *word-level sentiment*. In document-level sentiment, the word sentiment information is consistent with the labeled document. Let M_{ij} be the number of word W_i appearing in documents of sentiment category j, $j = 1, \ldots, c$ and c is the number of sentiment categories. If the prior sentiment information adopts the distribution, then W_i's sentiment distribution, S_i, can be written as $S_i = (M_{i1}, M_{i2}, ..., M_{ic})/\sum_j^c M_{ij}$. Alternatively, we can represent the sentiment information as the ratio between the positive category and negative category, which is written as $B_i = M_{i,pos}/\alpha M_{i,neg}$, where α is used to control the sentiment discrepancy. In word-level sentiment, no document labels are available. The prior sentiment information comes from the word sentiment dictionary. Generally, each word falls into one of the three categories $\{positive, neutral, negative\}$. We then represent the word-level sentiment information as

$$B_i = \begin{cases} 1/\alpha & positive \\ 1 & neutral \\ \alpha & negative \end{cases} \tag{2}$$

where $\alpha \in (0, 1]$ is the contrast ratio. The smaller α, the more significant the contrast is. In the next two sections, we describe how to incorporate document-level sentiment and word-level sentiment, respectively.

Incorporating Document-Level Sentiment

As stated previously, by only considering the ratio of co-occurrence probabilities as in Eq. 1, the learnt embeddings cannot reflect the sentiments of words. To incorporate the sentiment information, let us consider replacing $\frac{P_{ik}}{P_{jk}}$ with $\frac{P_{ik}}{P_{jk}} \cdot \frac{S_i}{S_j}$. Note that $\frac{P_{ik}}{P_{jk}} \cdot \frac{S_i}{S_j}$ is a vector and can be seen as a compound sentiment similarity and co-occurrence measure, where $\frac{P_{ik}}{P_{jk}}$ is the ratio of co-occurrence probability so as to keep the ability of capturing semantic meanings of words from documents, while $\frac{S_i}{S_j}$ is a measure of sentiment similarity so as to capture the sentiments from prior information. Thus, even if two words such as "like" and "dislike" have the same contexts so that $\frac{P_{ik}}{P_{jk}} \approx 1$, their sentiment distributions are different and, hence, correspond to different vector representations. With $\frac{P_{ik}}{P_{jk}}$ replaced by $\frac{P_{ik}}{P_{jk}} \cdot \frac{S_i}{S_j}$, Eq. 1 is rewritten as

$$F\left(\tilde{w_k}^T w_i s_i - \tilde{w}_k^T w_j s_j\right) = \frac{P_{ik} S_i}{P_{jk} S_j} \tag{3}$$

where s_i and s_j are two vectors of length c to match the size of $\frac{P_{ik} S_i}{P_{jk} S_j}$ and are also to be learnt. Similarly, we can also replace $\frac{P_{ik}}{P_{jk}}$ by $\frac{B_i}{B_j}$, then s_i, s_j are scalars.

There is no difference between target word and context when we select them arbitrary in the same corpus, and the roles of target word and context word are symmetrical. To keep this consistently, on the right hand of Eq. 3, when exchanging $w \leftrightarrow \tilde{w}$, it requires F to be a homomorphism between group $(R, +)$ and $(R_{>0}, \times)$, i.e., [23]

$$F\left(\tilde{w}_k^T w_i s_i - \tilde{w}_k^T w_j s_j\right) = \frac{F\left(\tilde{w}_k^T w_i s_i\right)}{F\left(\tilde{w}_k^T w_j s_j\right)} \tag{4}$$

So Eq. 3 is modified into,

$$\frac{F\left(\tilde{w}_k^T w_i s_i\right)}{F\left(\tilde{w}_k^T w_j s_j\right)} = \frac{P_{ik} S_i}{P_{jk} S_j} \tag{5}$$

And we have,

$$F\left(\tilde{w}_k^T w_i s_i\right) = P_{ik} S_i \tag{6}$$

The solution to Eq. 6 is $F(x) = exp(x)$, which leads to

$$\begin{aligned} \exp\left(\tilde{w}_k^T w_i s_i\right) &= P_{ik} S_i \\ \tilde{w}_k^T w_i s_i &= \log(P_{ik} S_i) \end{aligned} \tag{7}$$

Note that in Eq. 7, we incorporate the sentiment of the target word, i.e., S_i. We can also incorporate the sentiment of the context word, which gives

$$\tilde{w}_k^T w_i s_k = \log(P_{ik} S_k) \tag{8}$$

For simplicity of notations, we use $m = \{i, k\}$ to denote the two cases as

$$\tilde{w}_k^T w_m s_m = \log(P_{ik} S_m) \tag{9}$$

where $m = i$ means that the prior information is the target word sentiment, and $m = k$ means that the prior information is the context word sentiment. By using the fact that

$$P_{ik} = X_{ik} / \sum_{i=1}^{n} X_{ik}$$

and

$$S_m = (M_{m1}, M_{m2}, ..., M_{mc}) / \sum_{t=1}^{c} M_{mt},$$

we have

$$\tilde{w_k}^T w_i s_m = \log(X_{ik} M_m) - \log\left(X_i \sum_{t=1}^{c} M_{mt} 1^{1\times c}\right) \tag{10}$$

where

$$M_m = (M_{m1}, M_{m2}, ..., M_{mc})^{(1\times c)}$$

and

$$X_i = \sum\nolimits_{k=1}^{V} X_{ik}$$

with V being the vocabulary size. All the operations are element-wise. Because the term of $\log(X_i \sum_{n=1}^{c} M_{in} 1^{1\times c})$ is independent of word w_k, and has relation with the sentiment distribution, we can incorporate them into $b_i s_m$. The same applies to the word $\tilde{w}_k$ by adding the additional biases $\tilde{b}_k s_m$.

$$\tilde{w}_k^T w_i s_m = \log(X_{ik} M_i) - b_i s_m - \tilde{b}_k s_m \tag{11}$$

After applying the sentiment ratio B_m as the prior information, we have,

$$\tilde{w_k}^T w_i s_m = \log(X_{ik} B_m) - \log(X_i) \tag{12}$$

where $m = \{i, k\}$ is the same as in Eq. 9. Unlike the case in Eq. 10, term of $\log(X_i)$ is independent of sentiment information and word w_k. Hence, we apply the bias b_i on word w_i and $\tilde{b}_k$ on word $\tilde{w}_k$.

In text analysis, it is common to have words that appear few times in the entire dataset. These words could be non-important words or words that are misspelled, which generally introduce noise into the dataset. To reduce the effects of low-frequency words, we introduce a weight function $f(X_{ij})$ into the cost function as GloVe does:

$$f(X_{ij}) = \begin{cases} (x/x_{\max})^{\eta} & if\ x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{13}$$

here η is the attenuation coefficient, the empirical value is 0.75, and $x_{\max}$ should cover the number of the common words. With the weight function defined above, the cost function of incorporating document-level distributional sentiment on target word (**DLJT1**) is:

$$J = \sum_{i,k=1}^{V} f(X_{ik}) \sum_{c=1}^{C} \left(\tilde{w}_k^T w_i s_i + b_i s_k + \tilde{b}_k s_k - \log(X_{ik} M_i)\right)^2 \tag{14}$$

where $s_i \in \mathbb{R}^{1\times c}$. And the cost function of incorporating document-level distributional sentiment on context word (**DLJC1**) is:

$$J = \sum_{i,k=1}^{V} f(X_{ik}) \sum_{c=1}^{C} \left(\tilde{w}_k^T w_i s_k + b_i s_i + \tilde{b}_k s_i - \log(X_{ik} M_k)\right)^2 \tag{15}$$

where $s_k \in \mathbb{R}^{1\times c}$. The cost function of incorporating document-level sentiment ratio on target (**DLJT2**) is:

$$J = \sum_{i,k=1}^{V} f(X_{ik}) \left(\tilde{w}_k^T w_i s_i + b_i + \tilde{b}_k - \log(X_{ik} B_i)\right)^2 \tag{16}$$

where s_i is scalar. And the cost function of incorporating document-level sentiment ratio on context word (**DLJC2**) is:

$$J = \sum_{i,k=1}^{V} f(X_{ik}) \left(\tilde{w}_k^T w_i s_k + b_i + \tilde{b}_k - \log(X_{ik} B_k)\right)^2 \tag{17}$$

where s_k is a scalar.

Incorporating Word-Level Sentiment

Since word-level sentiment is also represented as sentiment ratio form, i.e., B_i in Eq. 2, the loss function is similar with incorporating document-level sentiment ratio of the word. Specifically, the objective function of incorporating word-level sentiment ratio of target word (**WLJT**) is

$$J = \sum_{i,k=1}^{V} f(X_{ik}) \left(\tilde{w}_k^T w_i s_i + b_i + \tilde{b}_k - \log(X_{ik} B_i) \right)^2 \tag{18}$$

And the objective function of incorporating word-level sentiment ratio of context word (**WLJC**) is

$$J = \sum_{i,k=1}^{V} f(X_{ik}) \left(\tilde{w}_k^T w_i s_k + b_i + \tilde{b}_k - \log(X_{ik} B_k) \right)^2 \tag{19}$$

Time Complexity

From Eqs. 14 to 17, besides Eqs. 18 and 19, the computational complexity mainly lies in the number of nonzero elements in matrix X and the category number c of the sentiment in document level when we use the sentiment distribution as the prior information. If the size of the vocabulary is $|V|$, and the corpus size is $|C|$, the computing time will not be worse than $c|V|^2$. When using the sentiment ratio or the sentiment value from the lexicons as the prior information, the computing time will not exceed $|V|^2$. The scales of all models mentioned above are no worse than $O(|V|^2)$. To find out the tight bound of the computing complexity, assume that the frequency rank of word pair r_{ij} in word co-occurrences follows the power-law function $X_{ij} = k/r_{ij}^{\beta}$, where $k = r_{max}^{\beta} = |X|^{\beta}$.

The actual computing complexity in document level with the sentiment distribution as prior information is proportional to the product between sum X and category number c, If we could find out the relation between $|C|$ and $|X|$ with the sentiment distribution as prior information, then both of tight bound of computing complexity in this model and computing complexity of other cases will be proven since $c = 1$.

$$|C| \sim \sum_{ij} X_{ij} M_i = c \sum_{r=1}^{|X|} \frac{k}{r^{\beta}} = kcH_{|X|,\beta} \tag{20}$$

where $H_{n,m}$ is the generalized harmonic number, and $H_{n,m} = \frac{n^{1-m}}{1-m} + \zeta(m) + O(n^{-m})$, *i.e.*, $s > 0, s \neq 1$, where $\zeta(m)$ is Riemann zeta function. Hence, we have

$$|C| \sim \frac{c|X|}{1-\beta} + c\zeta(\beta)|X|^{\beta} + O(1) \tag{21}$$

If $\beta > 1$, $c\zeta(\beta)|X|^{\beta} \sim 1$, then we have $|X| = O(|C|^{1/\beta})$, while $\beta < 1$, $c\zeta(\beta)|X|^{\beta} \sim 0$, then we have $|X| = O(|C|)$. We can find that there is no influence for category number c on computing complexity, and it is much better than the worst case $O(V^2)$ when $\beta > 0.5$ and $\beta \neq 1$. The same applies to the other models presented in this paper.

Experimental Results

In this section, we conduct experiments to evaluate the quality of the embeddings learnt with the proposed framework and which way of incorporating sentiment works best. A good word representation that captures sentiment should be good at word sentiment analysis and sentence classification. Thus, we first compare the proposed framework with state-of-the-art word embedding algorithms on word sentiment analysis. We also conduct experiments to investigate how the parameter α would influence the performance. Further experiments are conducted to evaluate the performance of the proposed framework for sentence classification.

Datasets

The embedding in "Sentiment Analysis for Words" is learnt from Stanford Sentiment Treebank (SST) [29], and the embedding in "Sentiment Analysis of Sentence" is learnt from MovieReview1 [21]. Apart from SST, dataset of MovieReview2 [22] is also used for the sentence classification. The labels in each dataset can be used as document-level sentiments. There are various standard sentiment lexicons such as MPQA, NRC, and SenticNet that can be used to obtain word sentiments. However, none of the lexicons is large enough to cover all the words in the vocabulary. Thus, the best way is to build document-level sentiment dynamically based on words in need. Specifically, we first manually labeled core seed dictionary. When it comes to a word which is not in the seed lexicon, we will get its synonym set from the Urban dictionary[1] and Youdao[2] at the same time. We then seek the intersection $I = \{I_1, I_2, I_3\}$ between the seed words and the synonym set, where I_1 denotes the positive word number, I_2 is the neutral word number and I_3 stands for the negative word number. The sentiment of the word is neutral when I is empty. Otherwise, the sentiment of the word will be as follows:

$$B = \begin{cases} 1/\alpha & I_1 \geq \max\{I_2, I_3\} \\ 1 & I_2 > \max\{I_1, I_3\} \\ \alpha & I_3 > \max\{I_1, I_2\} \end{cases} \tag{22}$$

[1]For example, we extract the synonym of word 'like' from page http://www.urbandictionary.com/define.php?term=like

[2]For example, we extract the synonym of word 'like' from the page of http://dict.youdao.com/search?q=like

where $\alpha \in (0, 1]$ is the contrast ratio. With this method, we can construct the word-level sentiment dictionary in the corpus of SST [29].

Sentiment Analysis for Words

In this subsection, we conduct experiments of sentiment analysis for words. We compare the performance of the proposed framework with state-of-the-art word embedding algorithms, i.e., CBOW [17], GloVe [23], and SE-HyRank [30]. Specifically, we first use the SST to train word embeddings. For DLJC1, DLJC2, DLJT1, DLJT2, we use document-level sentiment. For WLJC and WLJ2, we use the word-level sentiment constructed dynamically as describe above. For each word embedding algorithm, we first learn word embeddings from the SST [29]. For all models, we empirically set the vector size of the word embedding to 50, the learning rate to 0.5, and use the AdaGrad [7] to optimize the cost function. After that, we train classifiers to perform sentiment analysis of words with the learnt word vector representation as features. We use NRC and MPQA as the ground truth of the word sentiments. Note that the word-level sentiment are constructed dynamically and does not use any information from NRC and MPQA. We use the classifiers as Adaboost, support vector machine (SVM), and Naïve Bayes (NB) classifier and N fold cross validation are used to perform the classification test. The results with $N = 5$ and $N = 10$ are reported in Table 1. From the table, we make the following observations:

- Among our proposed models, DLJT1 and DLJC1 have the worst results, which clearly shows that the sentiment distribution as the prior information does not bring much improvement in sentiment analysis, and it is hard for word embeddings to imply the prior information by adding it in the distribution format.
- DLJT2, DLJC2, WLJT, and WLJC outperform other word embedding algorithms such as GloVe, Skip-gram and CBOW, which suggests that, by incorporating sentiment information into the learning process, word embeddings can capture more meaningful sentiment information.
- Comparing DLJT2, DLJC2 with WLJT and WLJC, we find that DLJT2 and DLJC2 slightly outperform WLJT and WLJC, respectively. This is because the label information is more reliable than the word-level sentiment constructed dynamically using the seed lexicons. However, the word-level sentiment is still useful for learning better word embedding as WLJT and WLJC significantly outperform GloVe.

Table 1 Results of the word sentiment validations

Embedding	N-Fold=5			N-Fold=10		
	Adaboost	SVM	NB	Adaboost	SVM	NB
	NRC					
CBOW	62.36%	61.93%	61.22%	62.46%	61.86%	61.33%
GloVe	60.59%	60.29%	60.42%	60.52%	60.22%	60.25%
SE-HyRank	54.57%	55.07%	50.28%	54.57%	55.07%	51.56%
DLJT1	58.82%	59.08%	59.32%	58.88%	58.28%	59.08%
DLJC1	57.34%	58.45%	59.79%	57.67%	58.51%	59.92%
DLJT2	*64.77%*	*64.92%*	64.94%	*64.67%*	*64.94%*	64.91%
DLJC2	64.27%	64.70%	*64.97%*	64.40%	64.87%	*64.97%*
WLJT	64.23%	63.43%	61.22%	64.15%	64.12%	62.49%
WLJC	64.10%	64.03%	61.32%	63.99%	64.06%	63.26%
	MPQA					
C&Bow	65.18%	65.82%	64.89%	65.43%	65.57%	64.82%
GloVe	64.11%	64.00%	63.64%	64.07%	64.07%	63.68%
SE-HyRank	60.89%	59.78%	60.79%	60.75%	60.79%	59.21%
DLJT1	63.21%	64.36%	63.43%	63.29%	64.47%	63.50%
DLJC1	62.36%	62.68%	63.54%	62.25%	62.39%	63.43%
DLJT2	*67.82%*	*68.75%*	67.82%	67.68%	*68.39%*	67.82%
DLJC2	67.68%	68.43%	*67.97%*	*67.89%*	*68.39%*	*67.93%*
WLJT	64.57%	64.64%	63.43%	65.53%	65.92%	64.36%
WLJC	66.32%	65.25%	63.43%	66.10%	65.85%	64.50%

Data in italics are best results

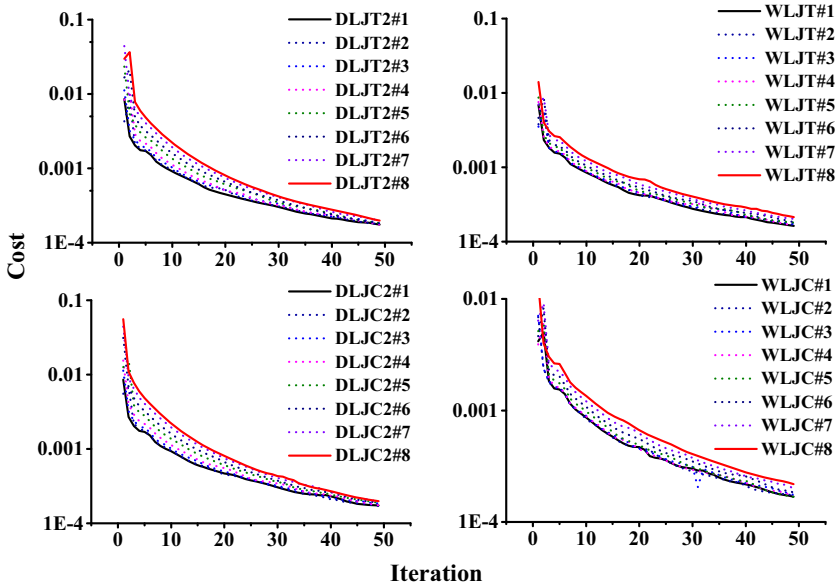

Fig. 1 The cost in each model with different α

Effects of α

In "Incorporating Word-Level Sentiment" and "Datasets", the contrast ratio α plays an important role representing prior sentiment information. In order to find out how the quality of word embedding changes with different α, we vary the value of α as {1, 0.75, 0.5, 0.25, 0.1, 0.05, 0.02, 0.01} and learn word embeddings. When $\alpha = 1$, we call DLJT2 as DLJT2#1. When $\alpha = 0.75$, we call call DLJT2 as DLJT2#2, so on so forth, and we call DLJT2 as DLJT2#8 when $\alpha = 0.01$. The same applies to DLJC2, WLJT and WLJC.

The objective function values of DLJT2, DLJC2, WLJT, and WLJC, in each training iteration are shown in Fig. 1, where we can see that α is inversely rational to the costs directly. From Eqs. 2 and 22, the smaller the value α, the larger the distance between positive and negative is. Furthermore, it takes more time to train the long distance value in the loss function. Hence, α should not be too small for training time efficiency.

To find out the influence of α on the word sentiment analysis, we vary the value of α as {1, 0.75, 0.5, 0.25, 0.1, 0.05, 0.02, 0.01} and learn embeddings for word sentiment analysis. The comparison results are shown in Fig. 2, where the blue lines are the results of incorporating word-level sentiment, and the red lines are for document-level sentiment. Obviously, the results of sentiment analysis accuracy with the document-level sentiment is much better than that using word-level sentiment. Generally, the accuracy increases with the decrease of α. When α is smaller than 0.05, most of the lines are flat. Based on the results in Figs. 1 and 2, a value of α within the range [0.01, 0.05] can generally achieve better results with both word-level and document-level sentiment.

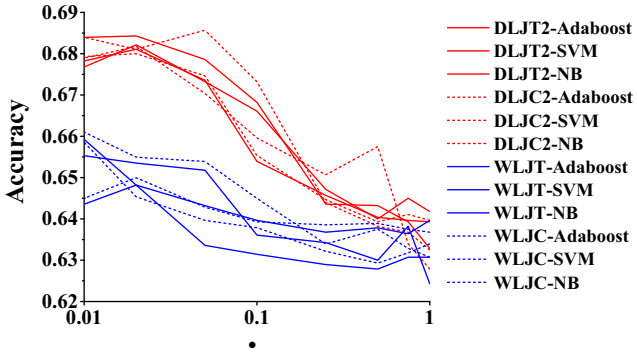

Fig. 2 Sensitivity of α on DLJT2, DLJC2, WLJT, and WLJC with word sentiment analysis using Adaboost, SVM and NB

Sentiment Analysis of Sentence

In this subsection, we further conduct sentiment analysis to evaluate the quality of word embedding learnt by DLJT2, as it has the best performance among all the models proposed for word sentiment analysis. Three datasets MovieReview1 [21], SST [29], MovieReview2 [22] are used for the evaluation, word embedding is learnt from the MovieReview1 [21], which ensures that the scale of the learnt words

Table 2 The coverage of the learnt words in other datasets

DataSet	Corpus_Num	Coverage
SST	13799	68.58%
MovieReview2	14696	64.99%

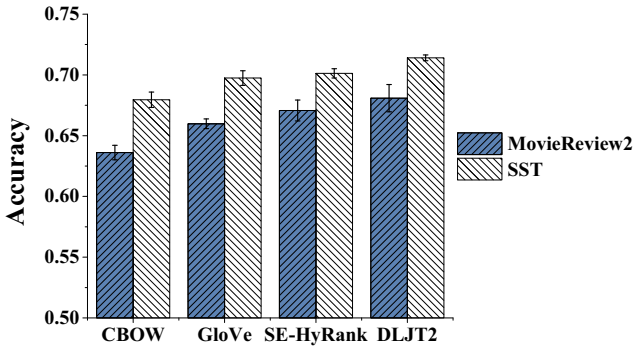

Fig. 3 The accuracy of the sentiment analysis by using different word embedding as the initial weight

covering at least 60% of the other datasets, the details are shown in Table 2. The dataset of MovieReview2 [22] and SST [29], which are used to do the sentiment analysis with the pre-trained embeddings, are all split into 80 and 20% sets and the former set is used for training.

The learnt embeddings are used as input to neural networks; the words that are absent from the learnt embeddings will be initialized randomly. In our experiments, we use a convolution neural network (CNN) [12] as classifier. To make comparisons, word embeddings from CBOW [18], GloVe [23], and SE-HyRank [30] are used as the baselines. Both the training episode and the architectures of CNN are the same for different word embedding algorithms. The results are reported in Fig. 3. From the figure, we can see that SE-HyRank has a better performance than that in word-level sentiment analysis, which is caused by the neural network structure that it uses (which only focuses on the abstract information). The proposed model DLJT2 outperforms the other word embedding algorithms, which suggests that word embedding learnt by DLJT2 can capture sentiments from different scales for better sentiment analysis.

Conclusion

In this paper, we investigated the problem of incorporating sentiment prior knowledge to learn meaningful word embeddings for sentiment analysis. We proposed a novel framework that can capture sentiment information of various types. Time complexity of the proposed framework is also comparable with the classical word embedding algorithm GloVe. Experimental results on word sentiment analysis and sentence sentiment analysis demonstrate the effectiveness of the proposed framework and show which prior knowledge is more effective. Parameter analysis was also performed to investigate the sensitivity of the proposed framework. In this work, we utilized label information from sentiment lexicons. As future work, we plan to explore how to use some strong emotional signals within the corpus for learning sentiment-specific word embeddings without using the labeled data or auxiliary lexicons.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Informed Consent Informed consent was not required as no human or animals were involved.

Human and Animal Rights This article does not contain any studies with human or animal subjects performed by any of the authors.

References

1. Bansal M, Gimpel K, Livescu K. Tailoring continuous word representations for dependency parsing. In: ACL (2). 2014. p. 809–815.
2. Bengio Y, Schwenk H, Senécal JS, Morin F, Gauvain JL. A neural probabilistic language model. J Mach Learn Res. 2003;3(6):1137–1155.
3. Cambria E, Das D, Bandyopadhyay S, Feraco A. A practical guide to sentiment analysis. Switzerland: Springer, Cham; 2017.
4. Cambria E, Poria S, Bajpai R, Björn S. SenticNet 4: A Semantic resource for sentiment analysis based on conceptual primitives. In: COLING; 2016. p. 2666–2677.
5. Chaturvedi I, Ragusa E, Gastaldo P, Zunino R, Cambria E. Bayesian network based extreme learning machine for subjectivity detection. J Franklin Inst. 2017. doi:10.1016/j.jfranklin.2017.06.007.
6. Collobert R, Weston J. A unified architecture for natural language processing: deep neural networks with multitask learning. In: International Conference, Helsinki, Finland, June; 2008. p. 160–167.
7. Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. J Mach Learn Res. 2011;12(Jul):2121–2159.
8. Harris ZS. Distributional structure. Synthese Language Library. 1954;10(2-3):146–162.
9. Hogenboom A, Bal D, Frasincar F, Bal M, de Jong F, Kaymak U. Exploiting emoticons in sentiment analysis Inproceedings of the 28th Annual ACM Symposium on Applied Computing; 2013. p. 703–710. ACM.
10. Hu X, Tang J, Gao H, Liu H. Unsupervised sentiment analysis with emotional signals. In: Proceedings of the 22nd international conference on World Wide Web; 2013. p. 607–618, ACM.
11. Huang EH, Socher R, Manning CD, Ng AY. Improving word representations via global context and multiple word prototypes. In: Meeting of the Association for Computational Linguistics: Long Papers; 2012. p. 873–882.
12. Kim Y. Convolutional neural networks for sentence classification. In: EMNLP. 2014.
13. Lin C-C, Ammar W, Dyer C, Levin LS. Unsupervised POS Induction with word embeddings. In: NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2015. p. 1311–1316.
14. Liu Y, Liu Z, Chua TS, Sun M. Topical word embeddings. In: Twenty-Ninth AAAI Conference on Artificial Intelligence. 2015.

15. Ma Y, Cambria E, Gao S. Label embedding for zero-shot fine-grained named entity typing. In COLING; 2016. p. 171–180, Osaka.
16. Majumder N, Poria S, Gelbukh A, Cambria E. Deep learning-based document modeling for personality detection from text. IEEE Intelligent Systems. 2017;32(2):74–79.
17. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. CoRR, arXiv:1301.3781; 2013.
18. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems. 2013;26:3111–3119.
19. Mnih A, Hinton G. Three new graphical models for statistical language modelling. In: International Conference on Machine Learning; 2007, p. 641–648.
20. Mohammad SM, Turney PD. Crowdsourcing a word-emotion association lexicon. Comput Intell. 2013;29(3):436–465.
21. Pang B, Lee L. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In: Proceedings of the ACL. 2004.
22. Bo P, Lee L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the ACL. 2005.
23. Pennington J, Socher R, Manning C. Glove: Global vectors for word representation. In: proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014.
24. Poria S, Cambria E, Bajpai R, Hussain A. A review of affective computing: From unimodal analysis to multimodal fusion. Information Fusion. 2017;37:98–125.
25. Poria S, Cambria E, Gelbukh A. Aspect extraction for opinion mining with a deep convolutional neural network. Knowl-Based Syst. 2016;108:42–49.
26. Poria S, Cambria E, Hazarika D, Vij P. A deeper look into sarcastic tweets using deep convolutional neural networks. In: COLING; 2016. p. 1601–1612.
27. Poria S, Chaturvedi I, Cambria E, Hussain A. Convolutional MKL based multimodal emotion recognition and sentiment analysis. In: ICDM; 2016. p. 439–448, Barcelona.
28. Rajagopal D, Cambria E, Olsher D, Kwok K. A graph-based approach to commonsense concept extraction and semantic similarity detection. In: WWW; 2013. p. 565–570, Rio De Janeiro.
29. Socher R, Perelygin A, Wu JY, Chuang J, Manning CD, Ng AY, Potts C. Recursive deep models for semantic compositionality over a sentiment treebank. In: proceedings of the conference on empirical methods in natural language processing (EMNLP), volume 1631, p. 1642, Citeseer. 2013.
30. Tang D, Wei F, Qin B, Yang N, Liu T, Zhou M. Sentiment embeddings with applications to sentiment analysis. Knowledge and Data Engineering, IEEE Transactions on. 2016;28(2):496–509.
31. Tang J, Qu M, Mei Q. Pte: Predictive text embedding through large-scale heterogeneous text networks. In: proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2015. p. 1165–1174, ACM.
32. Wang S, Tang J, Aggarwal C, Liu H. Linked document embedding for classification. In: CIKM. ACM; 2016.
33. Wang Y, Wang S, Tang J, Liu H, Li B. Unsupervised sentiment analysis for social media images. In: Proceedings of the Twenty Fourth International Joint Conference on Artificial Intelligence, IJCAI, 2015, Buenos Aires, Argentina; 2015. p. 2378–2379.
34. Wilson T, Wiebe J, Hoffmann P. Recognizing contextual polarity in phrase-level sentiment analysis. International Journal of Computer Applications. 2005;7(5):347–354.
35. Zhong X, Sun A, Cambria E. Time expression analysis and recognition using syntactic token types and general heuristic rules. In: ACL. 2017.
36. Zhou C, Sun C, Liu Z, Lau FCM. Category enhanced word embedding. Computer Science. 2015.
37. Zou WY, Socher R, Cer DM, Manning CD. Bilingual word embeddings for phrase-based machine translation. In: EMNLP; 2013. p. 1393–1398.