Embedding-based Feature Extraction Methods for Chinese Sentiment Classification

Sheng Zhang*, Hui Wang†, Xin Zhang‡, and Jiajun Cheng§, Pei Li¶, Zhaoyun Ding‖
*College of Information Systems and Management
National University of Defense Technology, Changsha, P. R. China
Email: zhangsheng@nudt.edu.cn
†Email: huiwang@nudt.edu.cn
‡Email: zhangxin@nudt.edu.cn
§Email: jiajun.cheng@nudt.edu.cn
¶Email: peili@nudt.edu.cn
‖Email: zyding@nudt.edu.cn

***Abstract*—Sentiment analysis, also known as opinion mining, seeks to figure out points of view from documents. Sentiment classification is a specific task of sentiment analysis that divides documents into positive and negative sentiment polarities according to the attitudes expressed. Feature extraction is a significant part of sentiment classification. Traditional feature extraction methods mine statistical information in documents but neglect semantic relationships between words, while some embedding methods successfully capture semantics but have difficulty in distinguishing sentiment polarities. In this paper, we propose three different sentiment specific models that take advantages of the statistical information in a document as well as the semantic relationships between words. Our models shed more light on the different roles of words in documents, assigning them different weights. Experimental results on three Chinese datasets illustrate that, in general, our models are superior to other models and provide state-of-the-art performance. Our models are efficient and have high stability.**

I. Introduction

Recent advances in electronic commerce have caused a growing number of people to pay attention to sentiment analysis, which is a useful tool in discovering the attitude of customers toward movies, commodities, or services. An essential and significant task in sentiment analysis is *sentiment classification*, which aims to divide a document into two sentiment polarities: positive and negative classes. Sentiment classification provides information to customers, operators, service providers, or manufacturers, and helps them promote and improve the quality of products and services. This task has been of increasing concern and plays an important role in modern society.

The procedure for sentiment classification generally consists of three steps [1], i.e. *preprocessing*, *feature extraction* and *classification*. Among these steps, feature extraction, or feature engineering, is the most critical [2]. Traditional feature extraction methods are used to mine statistical information in documents, such as Bag-of-Words models, n-grams models, lexicons-based models, and Part-of-Speech-based models. However, these models have difficulty in representing semantic and syntactic information, which means that two semantically similar words may have irrelevant features. For example, the words "excellent" and "great" express similar semantics, but have irrelevant features in these methods. In fact, semantic features are capable of revealing the implicit relationship between words and documents, which is beneficial in sentiment classification [3].

In recent years, embedding-based methods have been used to extract features from text given their excellent capability of semantic representation. Feature embedding methods, such as Word2Vec [4] and Doc2Vec [5], are designed to represent semantic information. These methods map words or documents into fixed-length vectors, in which semantically similar words or documents have vectors that are close in distance. Particularly, Doc2Vec yields state-of-the-art performance in sentiment classification on the English IMDB dataset [5].

Nevertheless, a limitation of Doc2Vec is that features extracted from documents are general, rather than sentiment specific [2], since Doc2Vec is an unsupervised learning algorithm. As a result, two semantically similar sentences may have opposite sentiment polarities. For example, "I like this mobile-phone, the screen is very big!" and "I dislike this mobile-phone, the screen is too big!" have similar vectors since "like"/"dislike" and "very"/"too" have a similar semantic role. However, this becomes a problem when using Doc2Vec for sentiment classification on some datasets.

Motivated by this limitation, we propose three sentiment specific feature embedding methods by combining traditional feature extraction methods and embedding-based methods. More specifically, position, Part-of-Speech, and Delta-TFIDF information are applied to Word2Vec to extract document features. Our method makes full use of lexical semantic relationship information and also takes advantage of statistical information in documents, which makes our algorithm more suitable for sentiment classification. Experimental results demonstrate that the proposed methods provide a significant improvement in sentiment classification and yield state-of-the-art performance.

In summary, the major contributions of the work presented in this paper are:

* By combining statistical information with feature em-

978-1-5386-1600-0/17 $31.00 © 2017 IEEE
DOI 10.1109/DSC.2017.46

bedding methods, we propose three different Word2Vec-based feature extraction methods, which take advantage of the information in documents. These methods overcome the semantic problems of traditional methods and the sentiment limitations of some existing embedding-based methods, such as Doc2Vec.

* Compared to previous models, our proposed methods reach state-of-the-art performance on the same datasets. The accuracy of our methods is up to 94%, with a relative improvement of 3% compared to other models.
* Once training word vectors, our models become highly efficient classification algorithms due to the low and fixed dimension of features.

The rest of this paper is structured as follows. Section II provides related work on sentiment classification and feature extraction. Section III compares five different feature extraction models and three types of classification algorithms that are used in the experiments. Section IV compares different methods and algorithms on three datasets. Finally, the conclusion is provided in Conclusion Section.

II. Related Work

Sentiment classification identifies the sentiment polarity of documents, including tweets and reviews of commodities and services, and has recently attracted much attention in research. Feature extraction is a critical part of sentiment classification.

Traditional feature extraction methods select proper words or use statistical information. Bo Pang et al. [6] used unigrams (Bag-of-Words) and bigrams to extract features, and machine learning techniques, such as Naive Bayes, Maximum Entropy, and Support Vector Machine for sentiment classification. This was a pioneering work in applying learning algorithms to sentiment classification. However, n-grams models take all words in a document into account, which can be time consuming.

An improvement to this approach is based on building sentiment lexicons, which aim to find sentiment indicators as features. The most important indicators of sentiment are sentiment words, also called opinion words. The list of sentiment words is known as a lexicon. Hatzivassiloglu et al. [7] proposed a method for building a sentiment lexicon based on word corpora. Taboada et al. [8] presented a lexicon-based approach for extracting sentiment from text. Lexicon-based methods may be very simple and effective, but, over time, updating the lexicon becomes unrealistic and strenuous. Also, different types of corpus may require different types of lexicons.

Part-of-Speech (POS) is a group of words or phrases that have similar grammatical properties. Kouloumpis et al. [9] and Gimpel et al. [10] used POS tagging to extract features from tweets, which was more efficient when compared to building lexicons. However, POS-based methods treat words with the same POS tag equally, which can introduce noise into classifiers. For instance, not all adjectives and adverbs are sentiment words, some are neutral words.

Other works use document frequency (TF-IDF) [11], mutual information [12], and the χ^2 statistic [13] as feature engineering methods. These methods perform well in certain sentiment classification tasks. Nevertheless, these methods have the drawback that they only mine statistical information from documents, rather than semantic and syntactic information, which means that the features for two similar words are irrelevant. In order to identify sentiment polarity, semantic and syntactic information needs to play an important role since they reveal a deep and implicit relationship for different words and documents.

With ongoing research in neural networks, embedding methods have been applied to many NLP tasks in recent years. A useful and often-mentioned type of embedding method is word embedding, also called distributed word representation. In this method, words are represented by fixed-length vectors, and these vectors have semantic and syntactic meanings. Bengio et al. [14] proposed a Neural Network Language Model (NNLM), which combined a Neural Network with Natural Language Processing. The model consisted of 3 layers: input layer, hidden layer, and output layer. Using NNLM, Bengio et al. successfully trained the word embedding. Collobert et al. [15] used a scorer to substitute the output layer in NNLM and successfully applied their models to several NLP tasks.

In order to accelerate computing, Mikolov et al. [4][16] simplified NNLM and proposed Word2Vec, which consists of two models, Continuous Bag-of-Words (CBOW) and Skip-gram. Word2Vec removes the hidden layer and uses a projection to reduce the calculations, so word vectors are trained easily and quickly. Word2Vec is popular and has been applied to many tasks. Taddy et al. [17] and Huang et al. [18] used Word2Vec for document classification tasks. Wang et al. [19] and Lin et al. lin2015learning used Word2Vec to extract a knowledge graph. Levy et al. [20] and Yang et al. [21] used Word2Vec as an implicit matrix factorization. However, word embedding methods give only the representation of a word, which is not the representation of a document. Le et al. [5] proposed the Doc2Vec model by adding an extra document vector to Word2Vec in order to obtain the feature of the document; the model was successfully used on an English IMDB movie review dataset and is a leader in state-of-the-art performance. Doc2Vec has been successfully used in various Natural Language Processing (NLP) tasks [22][23], including sentiment classification and information retrieval. However, Doc2Vec is more of a general feature extraction method than a sentiment specific method. It neglects some useful and important information, such as position, Part-of-Speech, and term frequency. This information is important in sentiment classification.

III. Methodology

In general, sentiment classification in documents consists of the three steps shown in Fig 1, which includes preprocessing, feature extraction, and classification. First, preprocessing tasks, such as word segmentation and eliminating stop words, are performed. Next, the documents are fed into a feature extraction model to obtain the vector of each document. After that,

Fig. 1. The procedure of sentiment classification.

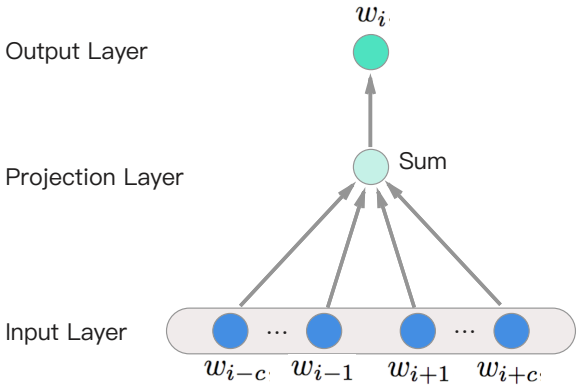


Fig. 2. The Continuous Bag of Word model.

a supervised classifier is used to classify the documents into two polarities: positive and negative classes.

In this section, the principles of word embedding methods (Word2Vec) are presented. Afterwards, Doc2Vec and other feature embedding methods are discussed. This is the main part of this section. Lastly, three classification algorithms that are used in our experiments are introduced.

A. Word Embedding Methods

Feature extraction influences the accuracy of the result and plays a vital role in sentiment classification tasks. A good feature extraction approach is capable of generating a satisfactory result even when using a weak classifier.

Word2Vec was first proposed by Miklov et al. [4], who were inspired by a Neural Network Language Model (NNLM) [14]. They simplified NNLM and reduced its computational complexity. Word2Vec consists of 2 models, the Continuous Bag-of-Words (CBOW) model and the Continuous Skip-gram model. Each model consists of three layers: input layer, projection layer, and output layer.

Fig 2 and Fig 3 show the CBOW model and Skip-gram model, respectively. Consider a certain word , w_i, and a fixed-size document window, $Context(w_i)$, made up of $w_{i-c}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+c}$, where $2 \cdot c$ is the window size.

Given $Context(w_i)$, the probability that the i^{th} word in the document is w_i equals $p(w_i|Context(w_i))$. Similarly, given a certain word, the probability of its surrounding words is $p(Context(w_i)|w_i)$. the target of the two models is to optimize the Log Maximum Likelihood Estimation (log MLE), as shown in Fig 2 and Fig 3, respectively:

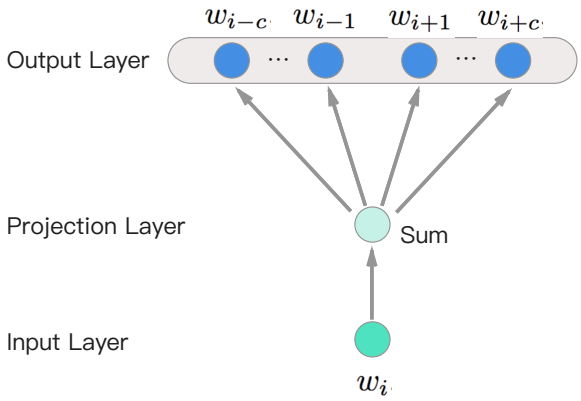


Fig. 3. The Continuous Skip-gram model.

$$\mathcal{L}(CBOW) = \sum_{w_i \in D} \log p(w_i|Context(w_i)), \quad (1)$$

and

$$\mathcal{L}(Skip-gram) = \sum_{w_i \in D} \log p(Context(w_i)|w_i), \quad (2)$$

where D is the document.

The model trains both neural networks and a vector matrix, which means that after training the Word2Vec models, word vectors can be obtained from neural networks.

B. Feature Extraction

Feature extraction plays a vital role in the procedure, since different methods affect the results in various ways. In this section, five embedding-based feature extraction methods are introduced that obtain the document feature. The first, called Doc2Vec, is the adaption of Word2Vec to a document. The remaining four methods combine Word2Vec with the information of a document. Specifically, Average Word2Vec is a blank control. Weighted Word2Vec, POS-based Word2Vec, and Delta TF-IDF Word2Vec use position, Part-of-Speech, and term frequency information of documents, respectively.

1) Doc2Vec: Doc2Vec is a document embedding method, shown in Fig 4 and Fig 5, that is similar to Word2Vec and was proposed by Le et al. [5] who added the document vector to the network. The document vector is treated the same as the word vector. After training the neural network, Doc2Vec is capable of computing the document vector, which is also the document feature.

Therefore, Eq (1) and Eq (2) can be adapted as:

$$\mathcal{L}(CBOW) = \sum_{w_i \in D} \log p(w_i|W(D), Context(w_i)), \quad (3)$$

and

$$\mathcal{L}(Skip-gram) = \sum_{w_i \in D} \log p(w_i, Context(w_i)|W(D)). \quad (4)$$

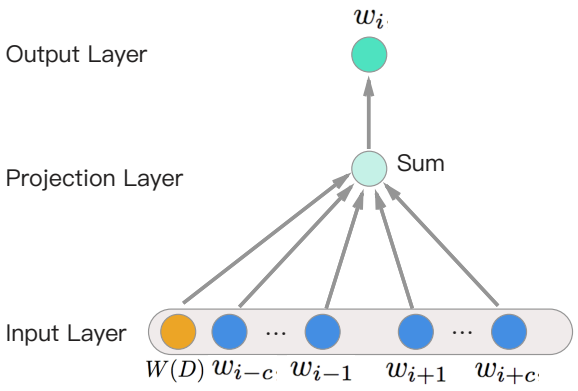


Fig. 4. Distributed Memory Model of Paragraph Vectors.

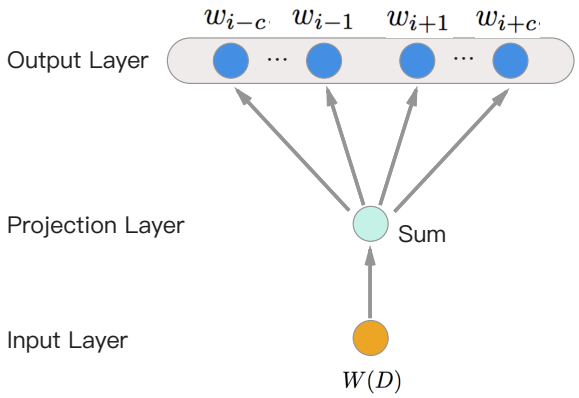


Fig. 5. Distributed Bag of Words version of Doc2Vec.

2) Average Vector: A simple approach for extracting the document feature is to obtain the average vector of each word, which is simply adding the vectors of all words and taking the average. This can be calculated as follows:

$$W(D) = \frac{1}{|D|} \sum_{i \in D} w_i, \tag{5}$$

Where $W(D)$ is the vector of document D, w_i is the vector of word i in D, and $|D|$ is the number of words in document D.

However, this method is simple and does not consider the importance of the word. Therefore, this method is used as a blank control and the following approaches improve on this by assigning different weights to different types of words based on their importance.

3) Weighted Vector: Based on intuition, the topic is typically at the front or end of a document, i.e., people are more likely to express emotions in these areas. This idea inspired us to weight different values of words in documents according to where the words are located. Therefore, words that appear in these areas should be assigned a greater weight than words in other areas. The calculation is given as:

$$W(D) = \frac{1}{|D|}\{C_1 \sum_{i \in D_1} w_i + (2 - C_1) \sum_{i \in D_2} w_i\}, \tag{6}$$

where D is the document, which is divided into two equal parts D_1 (mid part) and D_2 (beginning and end parts), and C_1 is a constant that satisfies $0 < C_1 < 1$, which is the weighting factor.

The domain of the weighting factor can be enlarged to $0 < C_1 < 2$. If $0 < C_1 < 1$ then the D_2 part is more important, otherwise, the D_1 part is more important. If $C_1 = 1$, the weighted Word2Vec model is equal to the Average Word2Vec model. By changing the value of C_1, we can confirm the hypothesis that the specified areas are of greater importance.

4) POS-based Method: Part-of-Speech tagging (POS tagging) assigns each word with a grammatical label. There are many Parts-of-Speech in Chinese. In order to simplify the problem, words are divided into four sets according to their POS tags: noun, verb, adjective/adverb, and others.

Before training word vectors, POS tagging can be done. After that, we extract the document feature using different weights of word vectors according to their POS tags.

Suppose that in document D, the number of nouns, verbs, adjectives/adverbs and other words are N_1, N_2, N_3, N_4, respectively, and the sets containing the words are S_1, S_2, S_3, S_4. In order to simplify the problem, we only lay emphasis on one type of word (S_1,S_2 or S_3) and we do not consider many structural words or stop words in S_4. Document features can be calculated as:

$$W(D,k) = \sum_{i \in S_k} \frac{C_2 \cdot w_i}{\sigma} + \sum_{i \in D \cap i \notin S_k} \frac{(2 - C_2) \cdot w_i}{\sigma}, \tag{7}$$

where k is the type of word that is emphasized. $W(D,k)$ is the weighted document vector with respect to the Part-of-Speech word set S_k, and $\sigma = \sum_{i \in S_k} C_2 + \sum_{i \in D \cap i \notin S_k} (2 - C_2)$.

We emphasize only one kind of Part-of-Speech word, increasing (or decreasing) the weight of those words.

Using weighting factor C_2, which satisfies $0 < C_2 < 2$, the model increases or decreases the weight of certain words in S_k. More specifically, for a POS set k, if $0 < C_2 < 1$, the weights of words in set k are lightened, and if $1 < C_2 < 2$, the weights are strengthened. If $C_2 = 1$, the POS-based Word2Vec model is equal to the Average Word2Vec model.

5) Delta TF-IDF: TF-IDF is used to weight words according to their occurrence in corpus. This idea has been widely used in many NLP tasks such as Information Retrieval (IR). By taking sentiment polarity into consideration, vectors can be weighted in a different way. Words that are more likely to occur in one sentiment polarity should be given more weight, e.g., “good” and “excellent” are more likely to occur in positive sentiment than computer. In this situation, “good” and

"excellent" should be assigned more weight than "computer" in terms of positive polarity.

Delta TF-IDF, proposed by Martineau [11], is based on Bag-of-Words by multiplying the TF-IDF score with the word count. However, it neglects information on semantic meaning. The method was adapted by using word vectors and word count; therefore, this model makes the best use of statistical information and semantic information. The equation is given as:

$$V'_{i,D} = C_{i,D} * \log_2 \frac{|P|}{P_i} - C_{i,D} * \log_2 \frac{|N|}{N_i}, \tag{8}$$

where i indicates the i^{th} word in document D, $C_{i,D}$ is the number of times word i occurs in document D. $|P|$ and $|N|$ are the number of positive and negative documents, respectively, and P_i and N_i are the word counts of i in the positive and negative documents, respectively.

If positive corpus and negative corpus are of equal size, the formula can be rewritten as:

$$V'_{i,D} = C_{i,D} * \log_2 \frac{N_i}{P_i}. \tag{9}$$

If $V'_{i,D} > 0$, then word i is more likely to occur in the negative document. On the contrary, if $V'_{i,D} < 0$, then word i is more likely to occur in the positive document. If $V'_{i,D} = 0$, then word i does not express sentiment. If $V'_{i,D}$ has a bigger absolute value, then word i is more important in expressing sentiment polarity.

In order to achieve good performance in vector space, Eq (9) can be regularized as:

$$V_{i,D} = \frac{V'_{i,D}}{\sqrt{\sum_{i \in D} V'^2_{i,D}}}. \tag{10}$$

After that, the document feature can be obtained by:

$$W(D) = \sum_{i \in D} V_{i,D} * w_i. \tag{11}$$

C. Classification Algorithms

After extracting the document features, the features are fed to the classifier, which divides them into two classes: positive and negative. In general, supervising algorithms such as Logistic Regression Classifier (LRC), K Nearest Neighbors (KNN), and Support Vector Machine (SVM) are used for classification. In this paper, two weak classifiers (LRC and KNN) and a relatively strong classifier (SVM) are chosen for the classification task.

1) Logistic Regression Classifier: Logistic Regression is a type of generalized linear model. The Binary Logistic Regression Classifier is commonly used in binary classification. The formula of the algorithm is:

$$f(X) = \frac{1}{1 + e^{-W \cdot X}}, \tag{12}$$

where X is the input, W is the weight, and $W \cdot X$ is the original linear model.

The model is simple and the data is labeled by a threshold, which is often 0.5. If $f(X) \leq 0.5$ then $y = 0$, otherwise $y = 1$.

2) K Nearest Neighbors: K Nearest Neighbors (KNN) uses information from the sample's nearest neighbors. The equation is given as:

$$y_i = \frac{1}{k} \sum_{j \in N_i} y_j, \tag{13}$$

where N_i is the neighbor set composed of sample i's k nearest neighbors in the training dataset, and y_i is the label of sample i.

3) Support Vector Machine: Support Vector Machine (SVM) is widely used in classification. An SVM algorithm finds a gap as large as possible that divides the labels of two different datasets. In addition, kernel tricks and soft-margin methods are used in SVM, so it performs well for nonlinear classification of datasets.

IV. Experiments and Results

This section consists of four parts, starting with a brief introduction to our datasets. The running environment is described in the second part. After that, the results of models on our datasets are presented and our proposed models are compared to previous models. At the end of this section, the parameters in our models are analyzed.

A. Datasets

Our data comes from NLPIR, which was also used by Zhang [24]. The data consists of three datasets, including reviews of hotels, books, and laptops. Details of these datasets are shown in Table I.

TABLE I
THE OVERVIEW OF DATASETS

Dataset Name	Pos	Neg
ChnSentiCorp-Htl-del-4000	2000	2000
ChnSentiCorp-Book-del-4000	2000	2000
ChnSentiCorp-NB-del-4000	2000	2000

The three datasets shown in Table I are balanced datasets, i.e., the count of positive and negative samples are equal. Usually, for a balanced dataset, accuracy is an appropriate metric.

B. Baseline

We compare embedding methods to traditional methods. A baseline model with Bag-of-Words (BOW) is built at the start. BOW is a method that constructs a dictionary and uses statistical information from the corpus. Each document can be presented as a fixed-size vector whose length is the size of the dictionary. BOW disregards the order of words in a document but maintains multiplicity, which means that each element of the vector is the occurrence of that word in the document.

In our experiment, a dictionary of our documents is built by adding words that appear in the documents. After that, each

document obtains a BOW vector according to the dictionary. The element in a BOW vector is the word count in the document.

C. Experimental Setup

The running environment and parameters are introduced in this subsection. Our programs are written in Python3 on a Macbook (2.6 GHz Intel Core i5 / 8 GB DDR3).

We use the Jieba package for segmentation and POS tasks. In order to obtain word vectors, the Python package Gensim [25] is used. We use both CBOW and Skip-gram to train 400-length vectors, and then we concatenate them into 800-length vectors. MinCount is the threshold of the word count, which we set to 1. The window size is set to 10. The method we use to compute the model is negative sampling, and the sample value is set to 10^{-3}. After obtaining the word vectors, the feature extraction methods discussed above are used to extract the vector of the document.

Similarly, Doc2Vec in Gensim gives us the document vector directly, and the parameter is set the same as above in Word2Vec.

The weighting factor C_1 in the Weighted Word2Vec model is set to $C_1 = 0.8$. The weighting factor C_2 and k in the POS-based Word2Vec model are set to $C_2 = 1.5, k = 3$, which means that adjectives and adverbs are given more emphasis. These parameters will be discussed later.

D. Results

In this subsection, experimental results are presented. The accuracies of models are given. Also, we compare results obtained by our models to those obtained by previous models.

Experiments on each extraction method and each classification algorithm are conducted separately. The Bag-of-Words model is used as the baseline.

Tables II through IV show the accuracy for each dataset. The first line of each table gives the baseline method, followed by the accuracy of the five feature extraction models in terms of three different classification algorithms. The highest scores are highlighted in bold.

Table II gives the results of the hotel review dataset. Delta TF-IDF performs best when using SVM classification, and the accuracy of Average Word2Vec is close to that of POS-based Word2Vec. Overall, SVM classification performs better than LRC and KNN.

Table III shows the accuracy of the book review dataset. Doc2Vec (using SVM) is approximately equal to POS-based Word2Vec, which reaches state-of-the-art performance when compared to other approaches. Using SVM is better than the other two algorithms.

Table IV gives the results of the notebook review dataset. Delta TF-IDF (using SVM) is again outstanding compared to the other models.

As shown in the three tables, the five feature extraction models produce relatively high accuracy, and most surpass the baseline when using KNN and SVM, but are inferior to baseline when using LRC, which means that KNN and SVM are more suitable for embedding features. As for classification algorithms, SVM is better than other algorithms, and Logistic Regression Classifier performs relatively well. However, KNN is unstable, which may be related to its definition of distance. Considering the good performance of SVM, we use SVM as the classifier when analyzing parameters later in this paper.

TABLE II
THE RESULTS OF CHNSENTICORP-HTL-DEL-4000

	LRC	KNN	SVM
Bag-of-Words (Baseline)	0.86759142	0.60025221	0.7629256
Doc2Vec	0.69331652	0.81967213	0.85977301
Average Word2Vec	0.84388398	0.77805801	0.87288777
Weighted Word2Vec	0.83556116	0.75863808	0.86506936
POS-based Word2Vec	0.84489281	0.80857503	0.87112232
Delta TF-IDF Word2Vec	0.83682219	0.84514502	**0.87515763**

TABLE III
THE RESULTS OF CHNSENTICORP-BOOK-DEL-4000

	LRC	KNN	SVM
Bag-of-Words (Baseline)	0.92336683	0.51005025	0.89572864
Doc2Vec	0.73517588	0.57462312	**0.93768844**
Average Word2Vec	0.88341709	0.6258794	0.93015075
Weighted Word2Vec	0.88291457	0.63768844	0.93442211
POS-based Word2Vec	0.88517588	0.62688442	**0.93668342**
Delta TF-IDF Word2Vec	0.89447236	0.87939698	0.92738693

TABLE IV
THE RESULTS OF CHNSENTICORP-NB-DEL-4000

	LRC	KNN	SVM
Bag-of-Words (Baseline)	0.89380531	0.67762326	0.8369153
Doc2Vec	0.68192162	0.84247788	0.85259166
Average Word2Vec	0.84450063	0.75676359	0.88166877
Weighted Word2Vec	0.84981037	0.75752212	0.88394437
POS-based Word2Vec	0.84753477	0.78457649	0.90012642
Delta TF-IDF Word2Vec	0.85562579	0.87661188	**0.90417193**

Another benefit of our models is that once training word vectors are established, we can quickly obtain the feature of each document. Also, due to the fixed and short length vectors of documents, the classifier is capable of training the dataset efficiently, and even some complex classifiers such as SVM can obtain a result in a short time.

Zhang et al. [24] also performed classification tasks on these datasets. They used several feature extraction methods such as bigram, sentiment lexicon, and substring, as well as several classification methods such as KNN and Rocchio. However, the feature methods they used were only based on statistical information instead of semantic information. The feature dimension in their models was so high that they had to perform feature reduction, which is a waste of time and costs thousands of seconds according to their paper.

Table V shows comparisons between our best results and their best results in terms of each dataset. Our results are superior to their results, with an improvement of about 3% at most. Since the models we proposed are more efficient, we can obtain results in a shorter time instead of thousands of seconds.

TABLE V
COMPARISON BETWEEN DIFFERENT MODELS

Dataset	Feature Extraction	Classification	Accuracy
ChnSentiCorp-Htl-del-4000	Bool-based Substring	Rocchio	0.871
	Delta TF-IDF Word2Vec(Ours)	SVM	**0.875**
ChnSentiCorp-Book-del-4000	TF-IDF Bigram	Rocchio	0.910
	POS-based Word2Vec(Ours)	SVM	**0.936**
ChnSentiCorp-NB-del-4000	TF-IDF Bigram	Rocchio	0.877
	Delta TF-IDF Word2Vec(Ours)	SVM	**0.904**

E. Parameters Analysis

In this subsection, we explore the effect of parameters for Word2Vec/Doc2Vec and the influence of the weighting factor in Weighted Word2Vec and POS-based Word2Vec. The control variable method is used to investigate parameter effects. By changing one parameter and fixing the others, we draw a line chart between parameters and accuracy scores. At the same time, we also compare different feature extraction methods or classification algorithms.

In our experiments, we primarily focus on the parameters for feature extraction methods. Five extraction methods are studied. As learned from Tables II through IV, the SVM classifier is superior to other classifiers in all three datasets. Therefore, it is reasonable to use the third dataset and the SVM classifier, since the other datasets or classifiers have similar properties or trends.

1) Parameters of Word2Vec and Doc2Vec: The parameters for Word2Vec (Doc2Vec) include vector length, window size, minimum count, and epoch times. In this subsection, we change the value of these parameters and rebuild the model to test how these parameters affect the results.

* Vector length: Vector length is the length of the word vector (document vector). The word vector is a fixed-length vector. In our experiments, 50, 100, 200, 300, 400, 500, 600, and 800 are used.
* Window size: Window size is the size of the model context. If the value is large, the model can learn more context patterns, but this is time consuming; otherwise, the model learns from a few words and little from context. This is always the trade-off between context knowledge and time cost.
* Minimum count: Minimum count is the minimum number of times that a word occurs in corpus. The model takes this into consideration.
* Epoch times: One epoch is the process of one forward pass and one backward pass for all training examples. Epoch times is the number of epochs. In general, more epoch times may lead to higher accuracy, but also higher time cost and variance.

In order to determine the influence of parameters for our embedding methods, we conduct experiments and change the parameters discussed above. Fig 6 shows the accuracy results.

Fig 6 consists of four subgraphs. The control variable method was used to investigate the influence of parameters discussed above.

Fig 6(a) shows accuracy in terms of the word vector length. Length varies from 50 to 800, while the other parameters are fixed. When the length goes beyond 200, the lines reach a steady level of accuracy.

Fig 6(b) shows accuracy in terms of the window size of Word2Vec. The curves increase with increasing window size and reach a plateau when the window size goes beyond 10.

Fig 6(c) shows the min word count. The lines fluctuate around a steady level.

Fig 6(d) shows the epoch times for training Word2Vec/Doc2Vec models. Accuracy values increase quickly before 10 epoch times, and then continue to increase slightly.

After analyzing the parameters of Word2Vec/Doc2Vec models, we find that NN-based (neural network based) models have great stability. Accuracy fluctuates around a steady level, so the values of parameters we used in our experiments are reasonable.

2) Parameters of Weighted Word2Vec: The weighting factor C_1 in the Weighted Word2Vec model is of great significance when extracting features from the corpus. It is necessary to consider how the value of C_1 influences the classification result.

In general, the topic of a document is at the beginning or end. The beginning and end parts are D_2 and the other parts are D_1, while D_1 and D_2 are divided equally in terms of word count. According to Eq (6), if $0 < C_1 < 1$, then words in D_2 are more important than those in D_1. If $1 < C_1 < 2$, then words in D_1 are more important. If $C_1 = 1$, then the model is equal to Average Word2Vec. By changing the value of C_1, we can confirm our hypothesis.

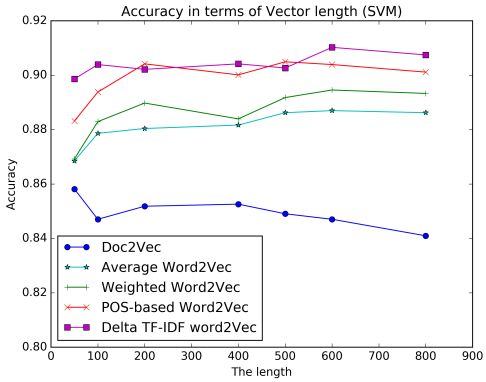
Fig 7(a) shows the accuracy of three classification algorithms in terms of different C_1 values. We determine that accuracy is higher when $0 < C_1 < 1$, compared to when $1 < C_1 < 2$, which is consistent with common sense. A sharp decrease occurs when $0.8 < C_1 < 1.2$. From the graph, we can draw a conclusion that Weighted Word2Vec surpasses Average Word2Vec to some extent, and that the weighting factor C_1 should be set to $0.5 < C_1 < 1$. In our former experiment, this parameter was set to $C_1 = 0.8$.

3) Parameters of POS-based Word2Vec: In the POS-based Word2Vec model, according to the Part-of-Speech, words are divided into four sets (S_1, S_2, S_3, S_4) which contain nouns, verbs, adjectives/adverbs, and other words, respectively.

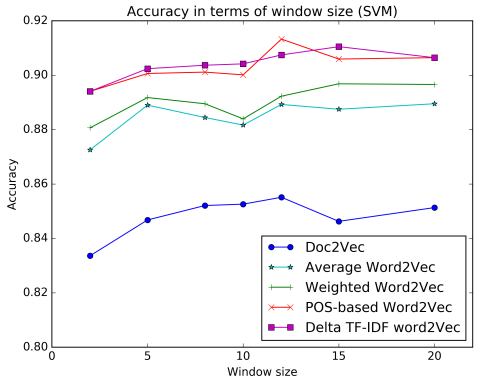
When users use different types of words, they emphasize these words. For example, users are more likely to use some adjectives or adverbs to express sentiment, so it is reasonable to apply more weight to words in S_3. The experiment shows that this speculation is true.

By changing the value of C_2, we can obtain the accuracy of SVM classification according to different k values (1, 2, or 3), which means that we apply more or less weight to S_1, S_2, or S_3. Since S_4 contains many structural words or stop words, we do not take $k = 4$ into account.

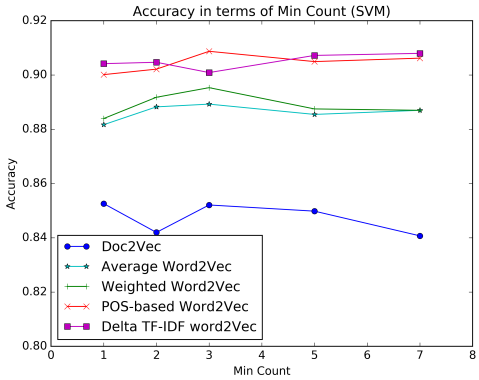
As shown in Fig 7(b), $C_2 = 1$ is an important cut-off point. If $0 < C_2 < 1$, words in S_k are given less weight and are



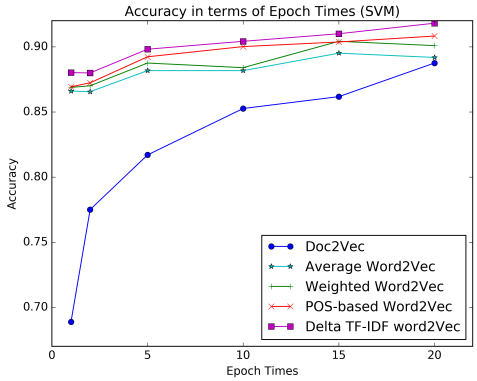
(a) Length.



(b) Window Size.

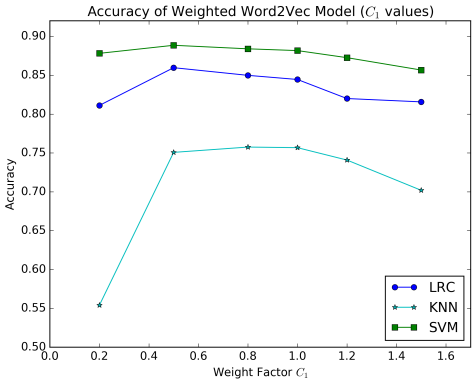


(c) Min Count.

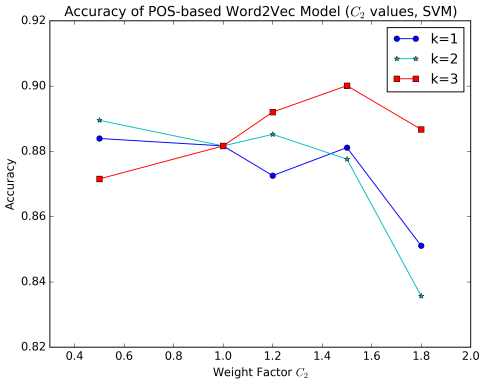


(d) Epoch Time

Fig. 6. Analysis of parameters of Word2Vec and Doc2Vec.



(a) Parameter C_1.



(b) Parameter C_2.

Fig. 7. Analysis of parameter C_1 and C_2.

relatively less important. In contrast, if $< 1C_2 < 2$, then words in S_k are given more weight and are relatively more important. On the left of the cut-off point, curve $k = 3$ is below the other two curves, but on the right, it surpasses the other curves, which means that weighting adjectives and adverbs plays an important role in sentiment expression. Therefore, laying more emphasis on words in S_3 and setting $C_2 = 1.5$ are good choices.

V. Conclusion

Word embedding methods have received significant attention and have been widely used in many Natural Language Processing tasks. In this paper, we proposed three feature extraction methods that combine embedding methods with position, Part-of-Speech, and term frequency information, which are more sentiment specific and therefore overcome the drawback of traditional feature extraction models and of some previous embedding methods. Our models are inspired by the idea of assigning different values to different words because different words play different roles in a document. Compared to other models, our models lead to better performance when using SVM and KNN classifiers on three datasets. In addition, compared to traditional models and previous work, our models yield state-of-the-art performance and are more efficient. The parameters of Word2Vec were carefully discussed and the experimental results showed great stability of the Word2Vec-based models. Also, the analysis of parameters in our models showed that the idea of assigning different weights to different words does work.

Although we obtain better performance, there is still room for improvement. The main drawback of Word2Vec-based models is that they neglect order information in a document, which may be crucial for some NLP tasks. In future work, we will attempt to add order information to our models in order to further improve performance.

Acknowledgment

The authors would like to thank the university, National University of Defense Technology, for providing a comfortable working atmosphere. The research is supported by National Natural Science Foundation of China (No.71331008) and (No.61105124).

References

[1] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 168–177.

[2] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 1555–1565.

[3] D. Zhang, H. Xu, Z. Su, and Y. Xu, "Chinese comments sentiment classification based on word2vec and svm perf," *Expert Systems with Applications*, vol. 42, no. 4, pp. 1857–1863, 2015.

[4] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *arXiv.org*, Oct. 2013.

[5] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," *arXiv.org*, May 2014.

[6] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment Classification using Machine Learning Techniques," *arXiv.org*, May 2002.

[7] V. Hatzivassiloglou and K. R. McKeown, "Predicting the semantic orientation of adjectives," in *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the association for computational linguistics*. Association for Computational Linguistics, 1997, pp. 174–181.

[8] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Computational linguistics*, vol. 37, no. 2, pp. 267–307, 2011.

[9] E. Kouloumpis, T. Wilson, and J. D. Moore, "Twitter sentiment analysis: The good the bad and the omg!" *Icwsm*, vol. 11, pp. 538–541, 2011.

[10] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith, "Part-of-speech tagging for twitter: Annotation, features, and experiments," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, 2011, pp. 42–47.

[11] J. Martineau and T. Finin, "Delta TFIDF: An Improved Feature Space for Sentiment Analysis." *ICWSM*, 2009.

[12] A. Bagheri, M. Saraee, and F. De Jong, "Sentiment classification in persian: Introducing a mutual information-based method for feature selection," in *Electrical Engineering*, 2013, pp. 1–6.

[13] H. Linsen, "Study on chinese text sentiment classification," 2014.

[14] Y. Bengio, Y. Bengio, R. Ducharme, R. Ducharme, P. Vincent, and P. Vincent, "A neural probabilistic language model," *The Journal of Machine ...*, vol. 19, no. 4, pp. 713–722, 2003.

[15] R. Collobert, J. Weston, L. Bottou, and M. Karlen, "Natural language processing (almost) from scratch," *The Journal of Machine ...*, 2011.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv.org*, Jan. 2013.

[17] M. Taddy, "Document classification by inversion of distributed language representations," *arXiv preprint arXiv:1504.07295*, 2015.

[18] C. Huang, X. Qiu, and X. Huang, "Text classification with document embeddings," in *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*. Springer, 2014, pp. 131–140.

[19] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes." in *AAAI*. Citeseer, 2014, pp. 1112–1119.

[20] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in Neural Information Processing Systems*, 2014, pp. 2177–2185.

[21] C. Yang and Z. Liu, "Comprehend deepwalk as matrix factorization," *arXiv preprint arXiv:1501.00358*, 2015.

[22] H. Liang, R. Fothergill, and T. Baldwin, "Rosemerry: A baseline message-level sentiment classification system," *SemEval-2015*, p. 551, 2015.

[23] J. Gutman and R. Nam, "Text classification of reddit posts."

[24] Y. Zhang, X. Xiang, C. Yin, and L. Shang, "Parallel sentiment polarity classification method with substring feature reduction," in *Trends and Applications in Knowledge Discovery and Data Mining*. Springer, 2013, pp. 121–132.

[25] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50.