Sentiment Aware Word Embedding Approach for Sentiment Analysis

Win Lei Kay Khine, Nyein Thwet Thwet Aung
University of Information Technology
Yangon, Myanmar
winleikkhine@uit.edu.mm, nyeinthwet@uit.edu.mm

Abstract

Nowadays, many business owners want to know the feedback of their products. If they get the feedback from customers, they can promote the quality of their products. So, Sentiment analysis has become a popular research problem to tackle in NLP field. It is the process of identifying whether the opinion or reviews expressed in a piece of work is positive, negative or neutral. We can apply sentiment analysis in brand monitoring, customer service, market research and analysis. Word embedding step is a problem in sentiment analysis of neural network models. Most existing algorithms for continuous word representation typically only model the syntactic context of words but ignore the sentiment of text. It is a problematic for sentiment analysis as they usually map words with similar syntactic context but ignore opposite sentiment polarity, such as good and bad, like and dislike. We solve this issue by proposing a method, sentiment-aware word embedding (SAWE). SAWE encodes sentiment information in the continuous representation of words by using (1) prediction the model and (2) ranking model. Finally, we evaluate our proposed method on IMDB movie review and twitter datasets, after that we prove our method outperform than other word embedding methods like word2vec and GloVe.

Keywords- Sentiment analysis, Natural Language Processing, Word Embedding, SAWE, Recurrent Neural Networks

1. Introduction

Nowadays, many business organizations want to promote their products in order to be successful. So, they survey about their products and use marketing strategies. It is expensive and time-consuming. If they use the sentiment application of their products, the above problem can be solved. If we do sentiment analysis, we first pass the data to the word embedding step.

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 and GloVe have been proven to facilitate various NLP tasks, such as word analogy, parsing, POS tagging, aspect extraction, etc... The majority of existing word embedding algorithms merely takes into account statistical information from documents. The representations learnt by such algorithms are very general and can be applied to various tasks. So, we propose sentiment-aware word embedding approach for sentiment classification task because it capture syntactic, semantic as well as sentiment information, unlike normal word embedding (word2vec and GloVe), which only capture syntactic and semantic information.

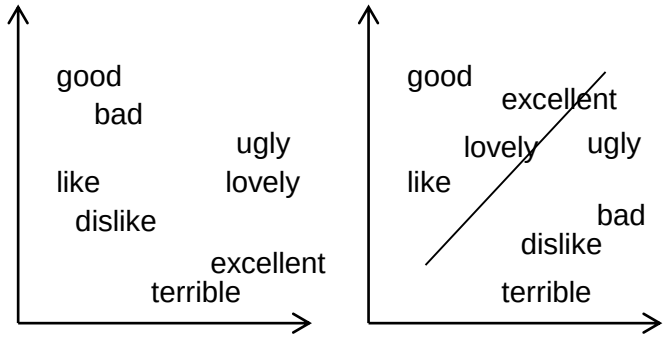


Figure 1. Normal word embedding (left) and sentiment-aware word embedding (right)

On the left of the figure(1) is normal word embedding which capture the syntactic context of words and the right one is capture and determine the sentiment of each word. So the positive and negative words are occupied separately in the vector space according to the polarity of the word. For example, the words “like” and “dislike” can appear in the same or similar context such as I *like* reading books or 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 embedding and, hence, learn better distributional representations for sentiment analysis.

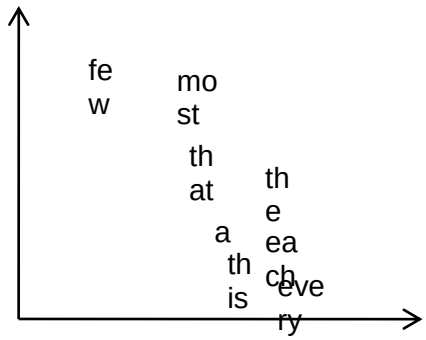


Figure 2. Word embedding for English determiners

The remaining paper is organized as follows. Section 2 describes the related works. Section 3 describes a methodology that is needed to implement sentiment classification. In Section 4, we present proposed system architecture and in section 5, we explain about datasets and experiment on these datasets. Finally, we conclude the paper in section 6.

2. Related Work

Word embedding is given to any method which converts words into numbers. We cannot directly feed text data to our machine learning or deep learning models. They won't work on strings of plain text. So, a natural language modeling technique like word embedding is used to map words to a corresponding vector of real numbers. Mikolov et al.(2013) introduce Continuous Bag-of-Words (CBOW) and Continuous Skip-gram, and release the popular word2vec toolkit. CBOW model predicts the current word based on the embeddings of its context words, and Skip-gram model predicts surrounding words given the embeddings of current word. [5].

Deep learning is the part of machine learning process which refers to Deep Neural Network. Neural Network is influenced by human brain and it contains several neurons that make an impressive network. It can have various numbers of nodes per layer, various numbers of hidden layers and weights connected between. The more layers a neural network has, the more complex model the network can learn. A neural network with multiple hidden layers is called Deep Learning [2]. Deep learning networks are capable for providing training to both supervised and unsupervised categories. It has been extensively applied in artificial intelligence field, computer vision, semantic parsing, and natural language processing many more [1].

In study by [3], the researchers proposed a novel Recursive Neural Deep Model (RNDM) to predict sentiment label based on recursive deep learning. In order to address the problem of little investigation on Chinese Sentiment analysis, they introduced a Chinese Sentiment Treebank and a powerful recursive deep model that can accurately predict the sentiment label on sentence level on movie review from social networks. The movie reviews were collected from http://movie.douban.com/. Finally, they reported that their RNDM obtains an accuracy of 90.8%, compared to NB (78.65%), ME (87.46%), and SVM (84.9%), so their RNDM achieves the highest accuracy in predicting binary sentiment label of sentence level.

The authors [4] proposed sentiment classification conducted on Japanese corpora. They trained and tested their sentiment classifier model, BiLSTM with Rakuten Merchant Review data. Their proposed can run without any dictionaries or features.

Another type of deep learning technique is Convolutional Neural Network (CNN). CNN consists of many layers that perform different functions. But CNN gives outstanding results in image processing and speech applications than text classification. The main problems of CNN are high computational cost and they need a lot of training data. And if you don't have a good GPU they are quite slow to train. And, if it is not having a good GPU, there will face a problem in training phase.

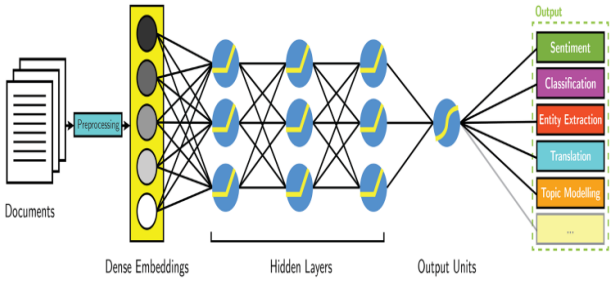


Figure 3.General deep learning based NLP

Figure 3shows the deep learning based NLP than any other classical NLP method. In our approach, we work with recurrent neural networks because it can handle more complex ways of connecting layers.

3. Methodology

In this section, we will discuss about word representations and popular techniques used in word embedding such as word2vec and GloVe.

3.1. Word Representations

Continuous word representation is commonly called word embedding, which attempt to represent each word as a continuous, low-dimensional and real-valued vector [6]. Word representation aims to represent aspects of word meaning. A straightforward way is to encode $w_i$as

a one-hot vector, whose length is a vocabulary size by 1 in the w_i^{th} position and zeros everywhere else. However, such one-hot vector word representation only encodes the indices of words in a vocabulary.

3.1.1 Word2Vec. Word2vec is not a single algorithm, but a combination of two techniques – CBOW (Continuous bag of words) and Skip-gram model. Both of these are shallow neural networks, which map word in the target variable which is also a word. Both of these techniques learn weights which act as word vector representations.

CBOW is learning to predict the word by the context. Or maximize the probability of the target word by looking at the context. And this happens to be a problem for rare words. For example, given the context *yesterday was really [...] day* CBOW model will tell you that most probably the word is *beautiful* or *nice*. Words like delightful will get much less attention of the model, because it is designed to predict the most probable word. This word will be smoothed over a lot of examples with more frequent words.

On the other hand, the skip-gram is designed to predict the context. Given the word delightful it must understand it and tell us, that there is huge probability, the context is yesterday was really [...] day, or some other relevant context. With **skip-gram** the word delightful will not try to compete with word beautiful but instead, delightful+context pairs will be treated as new observations.

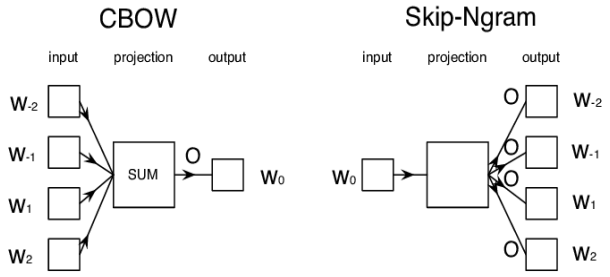


Figure 4.Continuous BOW a and Skip-Ngram approach used in word2vec

3.1.2 GloVe. An alternative approach for word embedding is called GloVe (Global Vectors) because the global corpus statistics are captured directly by the model. It is a model for distributed word representation. The model is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. It is developed as an open-source project at Stanford.

3.1.3 Sentiment Aware Word Embedding. The problem statement using word2vec and GloVe for word embedding is that they can't capture the sentiment information and they can only capture the syntactic and semantic information. So, the important feature for our work is to use Sentiment Aware Word Embedding (SAWE). SAWE can capture syntactic, semantic as well as sentiment information.

To implement word embedding in our system, we first describe standard context-based neural network methods for learning word embedding. Afterwards, we introduce our extension for capturing sentiment polarity of sentences which encode both sentiment and context level information. We then describe the integration of sentence level information for embedding learning. The discussion of the detail implementation of SAWE will explain in section 4.2.We implement our work with neural network layers, including *lookup*, *hTanh*, *linear* and *softmax*. For each neural layer, O_{layer}means the output vector.

Here, the proposed system builds sentiment-aware word embedding model in TensorFlow, an open source software library for high performance numerical computation. To work with TensorFlow, the Windows must be 64 bits-based Windows. We train our data on Window 7 64 bits, Core(TM) i7-4470, RAM 4.00 GB and require Python version 2.7 and above. We train and test the data on jupyter notebook in Anaconda Navigator.

4. Proposed System Architecture

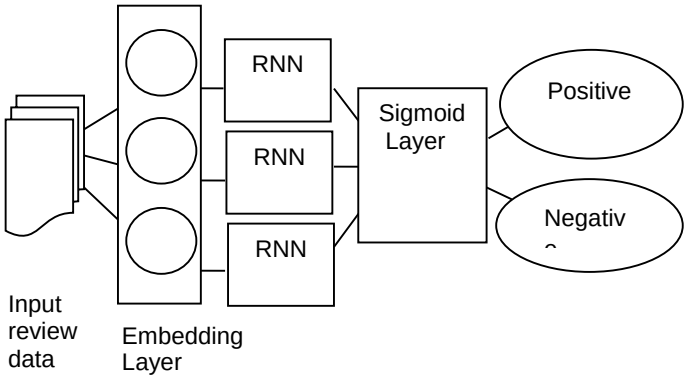


Figure 5.Overview of sentiment analysis using deep learning

According to the Figure 5, we firstly collected reviews texts data (Imdband Twitterdataset). Twitter data http://twitter.com by using Twitter API. Secondly, we need to pass words to an embedding layer and train up with SAWE model. Word embedding is not itself a deep learning technique, but it can turn raw text into a numerical form that deep nets can understand. For the word embedding, we used Keras in Embedding layer.

Keras is an open source neural network library. We run Keras on Tensorflow. We encode all words into word vectors. And then, we train Neural Network models on word vectors for classification. From the embedding layer, the new representations will be passed to Recurrent Neural Network (RNN).Finally, it goes to a sigmoid output layer. The sigmoid function takes any range real number and returns the output value which falls in the range of 0 to 1. Although there are many activation function, but we use sigmoid because it can predict if the text has positive or negative value for sentiment analysis. This is overview of my proposed system. But, my contribution is on sentiment-aware word embedding method on word embedding step.

4.1 Modeling Contexts of words

There are two parts in our proposed system. The first one is to model the context words and the second one is to model for sentiment polarity of the texts. In both of these sections, we will have to develop the prediction model and ranking model.

4.1.1 Prediction Model. In order to predict the contexts of words, we need to encode these contexts of words into word representation. This is called context prediction. Context prediction aims to predict the target word w_i based on its context words h_i. We need to predict surrounding words $h_i = \{w_{i-c}, w_{i-c+1}, \dots w_{i-1}, w_{i+1}, \dots w_{i+c-1}, w_{i+c}\}$. We build a prediction model analogous to the representative "context prediction" neural language model given by Bengio et al. [8]. They model the conditional probability $P(w_i | h_i)$ of predicting a target word $w_i$based on its contexts $h_i$by taking word embeddings as a parameter. The scoring function is a feed-forward neural network consisting of *lookup → linear→ hTanh→ linear→ softmax.*

Lookup layer also referred to as projection layer, which contains a lookup table $LT \in R^{d \times |V|}$ which maps each word to its continuously vector.

d= dimension of each word

V= vocabulary size

LT= lookup table

The lookup operation can be viewed as a projection function that uses a binary vector idx_i, which is zero in all positions except at the i^{th} index.

$$e_i = LT \,.\, idx_i \in R^{1*d} \quad (1)$$

After that, we concatenate the embedding of context words as the output of lookup layer.

$$O_{lookup} = [e_{i-c}; \dots e_{i-1}, e_{i+1} \dots; e_{i+c}] \in R^{1*d.2} \quad (2)$$

And then, the output of lookup layer is fed to a linear layer for dimension transformation is

$$O_{l1} = W_{l1} \cdot O_{lookup} + b_{l1,} \text{where} \quad (3)$$

W_{l1}= position-dependent weight

b_{l1}= bias of linear layer

O_{l1}= output vector of linear layer

In order to predict the probability of positive/negative polarity, we use *hTanh* (hard hyperbolic tangent) for its computational efficiency and effective in literature [9]. The output vector of hTanh is $O_{hTanh} \in R^{1*len}$

$$hTanh(x) = \begin{cases} -1 \; if \; x < -1 \\ x \; if \; -1 \le x \le 1 \\ 1 \; if \; x > 1 \end{cases} \quad (4)$$

The output layer is a softmax layer whose output length is vocabulary size. The probability of given sample from data D is defined as,

$$P(D|w,\theta) = \frac{\exp(f_\theta(w_i, h_i))}{\exp(f_\theta(w_i, h_i)) + k.\exp(f_\theta(w^n, h_i))} \quad (5)$$

The score function $f_\theta(w,h)$quantifies the compatibility between context $h_i$and target word w_i, which can be naturally defined as a feed forward neural network consisting of *lookup → linear → hTanh→ linear*. The input of lookup layer is the concatenation of the current word *w* and context words *h*. The output is a linear layer with output length as 1, which stands for the compatibility between context *h* and word *w*. We implement $P(D|w,\theta)$ with a *softmax*layer and maximize the log probability of the *softmax*for parameter estimation

Finally, the prediction for context of word is

$$loss_{cPred} = \sum_{w \in T} log\, P(D|w,\theta) \quad (6)$$

4.1.2 Ranking Model. Collobert and Weston[7] use a pairwise ranking approach to capture the contexts of words for learning wordembeddings. It holds the similar idea with noise contrastive estimation but the optimizing objective is to assign a real word-context pair (w_i, h_i) a higher score than an artificial noise (w^n, h_i) by a margin. They minimize the following hinge loss function, where *T* is the training corpora.

$$loss_{cRank} = \sum_{(w_i,h_i) \in T} max(0, 1 - f_\theta(w_i,h_i) + f_\theta(w^n, h_i)) \quad (7)$$

The scoring function $f_\theta(w, h)$ is achieved with a feed forward neural network. Its input is the concatenation of the current word $w_i$and context words h_i, and the output

is a linear layer with only one node which stands for the compatibility between *w* and *h*. During training, an artificialnoise $w^n$is randomly selected over the vocabulary under auniform distribution.

4.2 Calculation sentiment polarity

In this section, we present the approach to encode sentiment polarity of sentences in sentiment embeddings. We describe two neural networks including a prediction model and a ranking model to take considerations of sentiment of sentences.

4.2.1 Prediction Model. An illustration of prediction model with binary sentiment categories (positive and negative) is shown in Figure 2(a). It contains five layers, namely *lookup → linear→ hTanh → linear→ softmax*. The input is a fixed-lengthword sequence $\{w_{i-c}, w_{i-c+1}, \dots, w_i, \dots w_{i+c-1}, w_{i+c}\}$,where $w_i$is the current word and *c* is window size. Lookup,linear and *hTanh*layers are described in Section 3.2.1. Theoutput of *hTanh*layer is used as features to predict thepositive and negative probabilities of input.

To predict the probabilities of positive and negative categories, we feed *hTanh*to a linear layer to convert the vector length to category number *C*which is 2 in the binary classification case.The parameters of the second linear layer are $W1 \in R^{C \times len}$and $b1 \in R^{1 \times C}$. We then add a *softmax*layer as the output layer to generate conditional probabilities over positive and negative categories. Letf^g (t) $\in$ R ^{1*C} where

t= input t

C= number of sentiment polarity labels

For example, f^g (t) =[1,0] means sentence with positive polarity and f^g (t) =[0,1] means negative polarity. We use cross entropy between sentiment distribution and predicted distribution as the loss function of softmax layer. For the corpus T, the loss prediction for prediction model is defined as,

$$loss_{sPred} = -\sum_t^T \sum_{k=\{0,1\}} f_t^g(t).log(f_k^{pred}(t)) \quad (8)$$

4.2.2 Ranking Model. Now, we present an alternative of prediction model, which is a ranking model that outputs two real-valued sentiment scores for a word sequence with fixed window size. The basic idea of ranking model is that if the sentiment polarity of a word sequence is positive, the predicted positive score should be higher than the negative score. Similarly, if the sentiment polarity of a word sequence is negative, its positive score should be smaller than the negative score.

For example, if a word sequence is associated with twoscores[f_{pos}^{rank} ,f_{pos}^{rank}], then the values of [0.7, 0.1] can be interpreted as a positive case because the positive score 0.7 is greater than the negative score 0.1. Based on this consideration, we develop a neural network based ranking model, as illustrated in Figure 6 (b). As is shown, the ranking model is a feed-forward neural network consisting of four layers (*lookup → linear→ hTanh → linear→ softmax*.). Compared with the prediction model as shown in Figure 2 (a), the *softmax*layer is removed because it's objective does not require probabilistic interpretation.

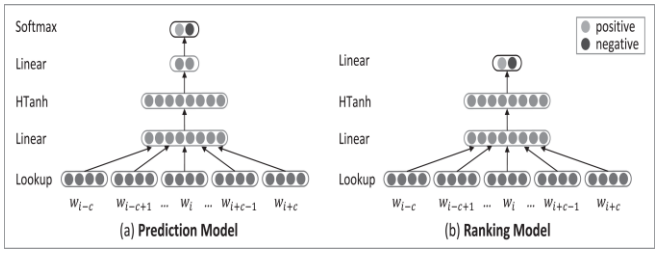


Figure 6. Layers in sentiment-aware word embedding

5. Experiment and Datasets

In this section, we explain about the data and compare the accuracy of difference word embedding methods on two datasets.

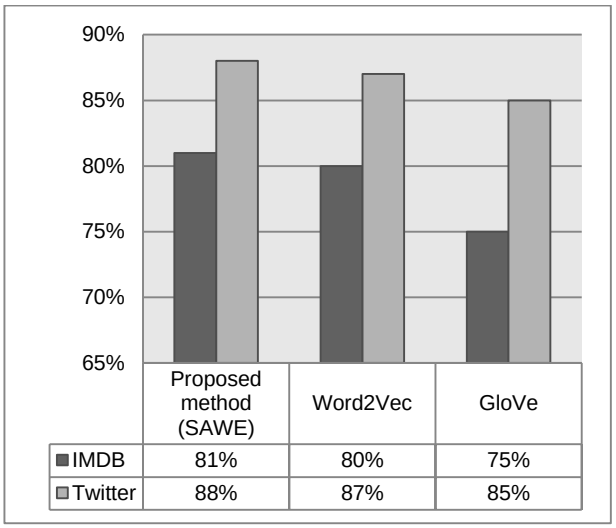
5.1 Data Collection

Table 1.Statics of the training datasets for sentiment-aware word embedding.

Dataset	#Positive	#Negative	#Total
IMDB	66,000	66,222	132,222
Twitter	637,728	665,432	1,303,160

We use two datasets for training the sentiment-aware specific word embedding separately. One is movie review data and the other is tweets.

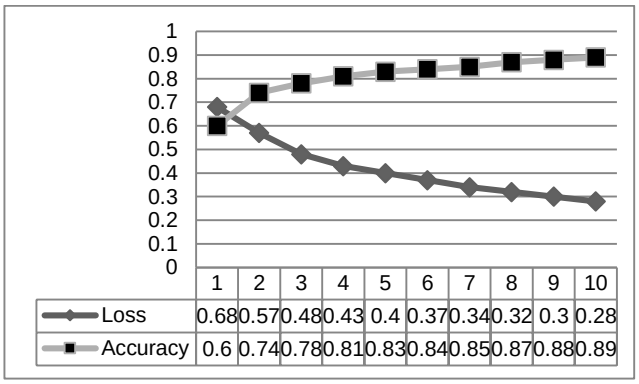
5.2 Result and Analysis



	Proposed method (SAWE)	Word2Vec	GloVe
IMDB	81%	80%	75%
Twitter	88%	87%	85%

Figure 7.Accuracy of positive/negative classification with different word embeddings

Accuracies for different word embedding techniques are shown in Figure 7. Our proposed method gets higher accuracy than word2vec and GloVebecause it can capture for both context and sentiment information of text.Because of having the training data of Twitter dataset is 10 times of IMDB dataset, the different accuracies can be seen. The more data we can train, the higher accuracy we will get.



	1	2	3	4	5	6	7	8	9	10
Loss	0.68	0.57	0.48	0.43	0.4	0.37	0.34	0.32	0.3	0.28
Accuracy	0.6	0.74	0.78	0.81	0.83	0.84	0.85	0.87	0.88	0.89

Figure 8.Loss and accuracy on IMDB dataset

Figure 8 explains about loss and accuracy on IMDB dataset. The loss function is one of the two parameters required to compile a model. Although there are many loss functions in neural networks, we use binary cross entropy in this system because it can classify on two classes.

6. Conclusion and Future Work

In this paper, we propose a sentiment-aware word embedding learning architecture for sentiment analysis. We encode the sentiment information into the continuous representation of words, so that it is able to separate good and bad to opposite of the spectrum. Finally, we prove that our method get higher accuracy than other word embedding techniques. Because of using sentiment-aware word embedding technique, it is effective for sentiment analysis than using normal word embedding techniques. For the future work, we aim to train on Burmese Text in different domain because language problem is also one of the challenges in sentiment analysis.

7. References

[1] Q.T.Ain, M.Ali, A. Riaz, A.Noureen, M.Kamran, B.Hayat and A.Rehman, "Sentiment Analysis Using Deep Learning Techniques: A Review", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 8, No.6, 2017, pp-424-433, 2017.

[2] P.Vateekul and T.Koomsubha, "A Study of Sentiment Analysis Using Deep Learning Techniques on Thai Twitter Data", 13th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2016.

[3] C.Li, B.Xu, G.Wu, S.He, G.Tian and H.Hao, "Recursive Deep Learning for Sentiment Analysis over Social Data", IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI), 2014.

[4] L.Nio and K.Murakami, "Japanese Sentiment Classification Using Bidirectional Long Short-Term Memory Recurrent Neural Network", The Association for Natural Language Processing, 2018.

[5] Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. "Efficient estimation of word representations in vector space."In Proceedings of the 1st international conference on Learning Representations (ICLR 2013).

[6] PengFu,ZhengLin,FengchengYuan,WeipingWang,DanMeng, "Learning Sentiment Specific Word Embedding via Global Sentiment Representation", Association for the Advancement of Artificial Intelligence, 2018, pp. 4808-4815.

[7] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning,"in Proceedings of the 25th international conference on Machine learning. ACM, 2008, pp. 160–167

[8] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," Journal of Machine Learning Research, vol. 3, pp. 1137–1155, 2003.

[9] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch,"Journal of Machine Learning Research, vol. 12, pp. 2493–2537, 2011.

.