ICIC Express Letters
Volume **11**, Number **6**, June **2017**

 ISSN 1881-803X
pp. **1133–1141**

A SELF-ADAPTIVE TEXT CLASSIFICATION METHOD BASED ON MULTIPLE WORD EMBEDDING AND NEURAL NETWORK

Tieli Sun[1,2], Yang Peng[1], Fengqin Yang[1,2], Hongguang Sun[1,2,*]
Na Fan[1] and Bangzuo Zhang[1]

[1]School of Computer Science and Information Technology
Northeast Normal University
[2]Key Laboratory of Intelligent Information Processing of Jilin Universities
No. 2555, Jingyue Ave., Changchun 130117, P. R. China
[*]Corresponding author: sunhg889@nenu.edu.cn

Received November 2016; accepted February 2017

Abstract. *Traditional text classifiers often rely on many human-designed features, which is complicated and inconvenient in practical applications. This paper proposes a novel neural network based text classification method that uses a self-adaptive mechanism to integrate various pre-trained word embedding. Firstly, the proposed method independently constructs feature vector from various word embedding with rich semantic information. Then, in order to distinguish the contribution of multiple word embedding in classification task and revise the influence of different embedding on the final text representation, we use Highway network to adjust the weight of feature vectors automatically. Finally, we combine these feature vectors and form the final text representation to classify. The experiment has been conducted on sentiment and news classification datasets; the results show that the method achieves better results with several benchmarks, and gets the similar performance to the state-of-the-art methods on English news classification task. The further contrast experiment also proves that the method has the ability of adaptively integrating multiple word embedding.*
Keywords: Text classification, Word embedding, Neural network

1. **Introduction.** The central problem of text classification is how to represent the documents, that is, how to construct the text feature vector. Traditional methods usually represent text based on bag-of-word model, and use features such as n-gram or other human-designed patterns [1-3]. However, these methods often face the problem of data sparsity, which heavily affects the classification performance.

In recent years, the deep learning method has demonstrated its perfect advantages in NLP, including learning distributed word representations [4] – also called "word embedding". Word embedding projects words from a high dimensional and sparse one-hot encoding onto a lower dimensional and dense vector space. The shift of embedding in the space represents the syntactic and semantic relations between words. This inspires us to combine word embedding by simple vector operations to generate efficient distributed text representation. Due to the difference of training methods and corpus, different word embedding often contains the word semantics information in different sides. It is very natural to explore combining different versions of word embedding and treating each of them as a distinct description of words. The proposed model can exploit the rich semantic information brought by different embedding to construct higher quality text representation and improve the classification performance. Some researches compare the differences between different word embedding [5,6], and show that different word embedding can achieve different performances in many NLP tasks. This inspires us to distinguish the contribution of multiple word embedding in classification task.

The neural networks also show its great performance in text classification, and many different network structures are used, such as recursion and convolution networks. Socher et al. proposed a series of recursion-based networks for text classification, such as Recursive Auto-Encoder (RAE) [7], Matrix-Vector Recursive Neural Network (MV-RNN) [8] and Recursive Neural Tensor Networks (RNTN) [9]; Kalchbrenner et al. used random initialization word vector as the input of a convolutional neural network and proposed a dynamic k-max pooling mechanism [10]; Kim initialized the input using pre-trained word embedding and proposed a CNN architecture with multiple filters (with a varying window size) and two channels of word embedding [11]; There are also some models with very deep structure, such as Character level CNN models (char-CNN) [12] and the Very Deep Convolutional Network (VDCNN) [13]. These deep models can learn high quality text representation via its deep architecture. Most of the NN-based classification models are very complex and need lots of training and testing time. Recently, a text classification model fast-Text [17] which has the same structure with the word embedding model CBOW can achieve state-of-the-art results on several large datasets with very little training time.

Although the existing text classification models have achieved remarkable performances, many of them have complex structures and a large number of hyper parameters. These models need huge training and testing time. All of these shortcomings limit their practical application. Also, the existing models often use single embedding as input, so that these models cannot exploit rich semantics brought by multiple pre-trained word embedding.

The motivation of our work is to leverage the rich semantics brought by different versions of pre-trained word embedding and use a self-adaptive mechanism to adjust the influence of multiple embedding on final text representation. The main contribution of this paper is a novel neural network based classification method. The proposed method can improve classification performance by integrating multiple versions of word embedding. It also uses a self-adaptive mechanism implemented by Highway network [15], which can automatically distinguish the contribution of multiple word embedding in classification.

The experiment conducts on sentiment and news classification task with several benchmarks. The experimental results show that our method can achieve better results. Moreover, the method is applicable to both Chinese and English text. In English news classification, the proposed method achieves similar results to the state-of-the-art approaches.

2. **Method.**

Figure 1 shows the structure of the proposed method using two word embedding. The method firstly generates text representation through two steps. (1) Use different word embedding independently to construct the corresponding text feature vector, and then adjust these feature vectors by Highway network so that the model can identify the important word embedding adaptively. In order to simplify the model and shorten the training time, we choose simple vector operations (sum, max and mean) to construct text feature vector, which is easy to be replaced by other more sophisticated methods (such as convolution). This means that our model is easy to extend. (2) Concatenate feature vectors to form the final text representation. The concatenation operation can handle vectors with different lengths, which conveniently integrate word embedding with different dimensions. Then, the text representation is fed into a fully-connected layer with dropout and Soft-max to calculate the probability of the text belonging to various categories.

2.1. **Text representation learning.**

We integrate multiple pre-trained word embedding to learn the text representation. The different embeddings often contain different words semantic information. We treat each of embeddings as a distinct description of words. Each of embeddings is used as an independent group to construct the corresponding feature vector. These feature vectors encode the different kinds of the text semantic information brought by corresponding word embedding.

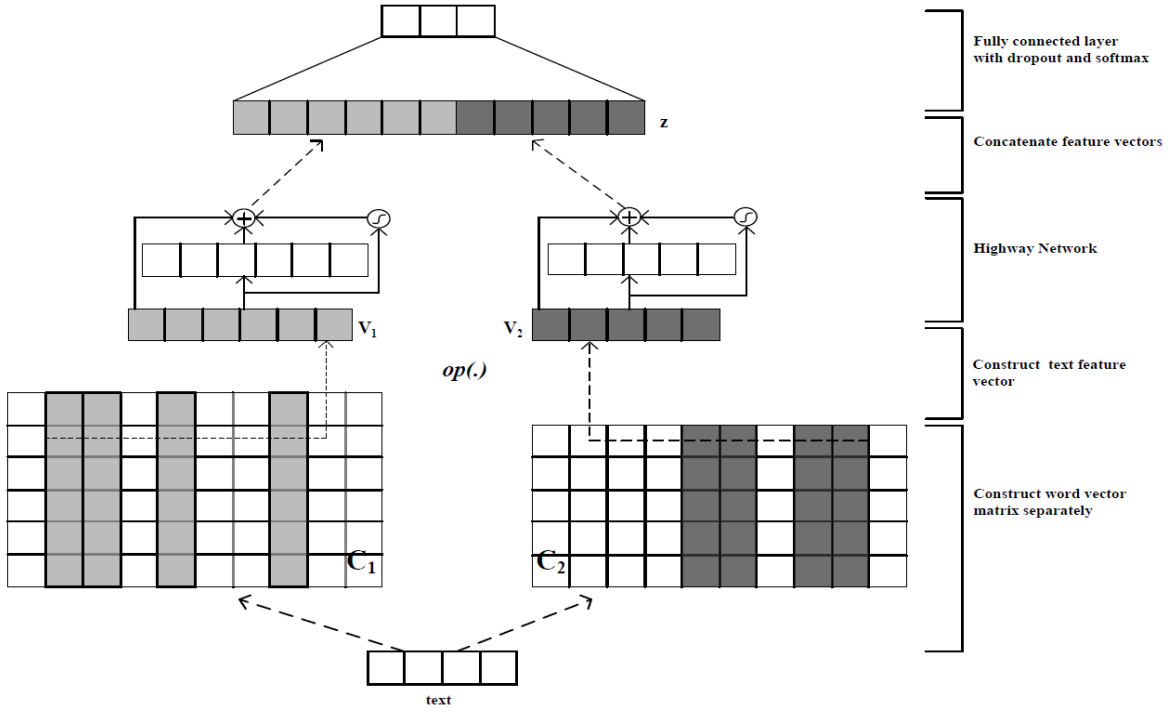

FIGURE 1. Model architecture with two-word embedding

The input layer of the model contains many lookup-tables, and each table corresponds to a word embedding. Let $C^i \in \mathbb{R}^{d_i * v}$ be the i-th lookup-table, where v is the size of word dictionary, and d_i is the dimension of the i-th embedding. Each column in table C^i represents the vector representation of a word in the dictionary. Before the training, each lookup-table is initialized with its corresponding word embedding, and then fine-tuned via back-propagation.

Suppose we use k word embedding, the input layer of the model also contains k lookup tables $(C^1, C^2, \ldots, C^k)$. For an input text $D = (w_1, w_2, \ldots, w_n)$ where w_i is the dictionary index of the i-th word in the text, the model looks up word vector matrix of the text from each lookup-table independently, and then combines each column of a matrix to obtain the corresponding text feature vector. The above process can be expressed by the following Formulas (1) and (2).

$$X_i = c^i_{w_1} \oplus c^i_{w_2} \oplus \cdots \oplus c^i_{w_n} \tag{1}$$

$$v_i = op\left(X^i\right) \tag{2}$$

where X^i is the word vector matrix constructed by lookup-table C^i, v_i is the text feature vector constructed by the i-th word embedding, and $op(.)$ represents the method of combining word vectors. There are a lot of ways to combine the word vectors, which make our method easy to extend. In order to ensure the computational efficiency, our model only uses some simple vector operation. However, it also can be replaced by more complex operations such as convolution.

$$v_i = c^1_{w_1} + c^2_{w_2} + \cdots + c^k_{w_n} \tag{3}$$

$$v_i = \left(c^1_{w_1} + c^2_{w_2} + \cdots + c^k_{w_n}\right)/k \tag{4}$$

$$v_i = \max\left(c^1_{w_1}, c^2_{w_2}, \ldots, c^k_{w_n}\right) \tag{5}$$

The k feature vectors $(v_1, v_2, \ldots, v_k)$ can be constructed from the above process. These vectors encode different kinds of text semantic information. We concatenate all vectors

to get the final distributed representation of the text:

$$z = [v_1, v_2, \ldots, v_i] \tag{6}$$

The use of concatenation operation makes our model can integrate embedding of different dimensions conveniently. The final text representation z contains k segments. Each segment is a text feature vector constructed from a different embedding, and represents a semantic description of the text. We expect the text representation contains the different semantic information brought by multiple word embedding, so that proposed method can effectively improve the classification performance.

2.2. **Highway network.** We use Highway network to distinguish the contribution of different versions of word embedding in classification task. Feature vectors generated by important embedding will retain in the final text representation z as many as possible, and unimportant vectors will be adjusted. As shown in Figure 1, the feature vector v_i is fed into a Highway network:

$$v_i^{'} = t \odot g(W_H v_i + b_H) + (1 - t) \odot v_i \tag{7}$$

where g is a nonlinearity, $t = \sigma(W_T v_i + b_T)$ is called transform gate, and $(1 - t)$ is called the carry gate. W_H, W_T are weight matrices, and b_H, b_T are bias vectors.

Highway network introduces the adaptive gating mechanism to manage the information flow of the network, so it can be used to implement a self-adaptive mechanism for identify important text feature vectors and automatically adjust the unimportant feature vectors. This mechanism can automatically distinguish the contribution of multiple word embedding and adjust the feature vectors generated by the embedding. The feature vectors constructed by important embedding will pass through Highway network directly from carry gate and do not produce any change basically. Those unimportant feature vectors will be blocked and adjusted by transform gate to reduce its impact of the final text representation. The contrast experiments are used to verify that the use of Highway network can improve classification performance.

3. **Experiments.** In order to evaluate the proposed model, the experiment is conducted on two classification tasks with three datasets.

3.1. **Datasets.** The datasets include Stanford Sentiment Treebank[1], FUDAN Chinese news corpus[2] and AG_NEWS English news corpus [12]. The classification task includes sentiment classification (SST1, finegrained classification of 5 sentiment labels; SST2, binary classification of positive and negative sentiment labels) and news classification (Chinese news classification on FUDAN datasets and English news classification on AG_NEWS datasets). The statistics of datasets are shown in Table 1.

Table 1. Statistics of the datasets after preprocessing

Data	C	L	N	$\|V\|$	Test	Lan
SST1	5	18	11855	17863	2210	EN
SST2	2	18	9613	16188	1812	EN
FUDAN	20	2118	20165	373656	9833	CH
AG_NEWS	4	44	127600	66478	7600	EN

C: Number of class labels. L: Average sentence length.
N: Dataset size. $|V|$: Vocabulary size.
Test: Test set size. Lan: Language of dataset.

[1]http://nlp.stanford.edu/sentiment/
[2]http://www.datatang.com/data/44139 and 43543

TABLE 2. Statistics of the pre-trained word embedding

Version	Training data	$\|V\|$	d	Language
Word2Vec_EH	Google News	4672123	300	English
Glove_300	Common Craw	2196017	300	English
Glove_50	Twitter	1193514	50	English
Huang	Wiki-EN	100232	50	English
Word2Vec_CH	Wiki-CH	697620	100	Chinese
Glove_CH	Wiki-CH	697620	50	Chinese

$|V|$: Number of words. d: Dimension of the embedding.

3.2. **Pre-trained word embedding.** In our experiments, we use 6 different versions of word embedding, and these were trained by three models (Word2Vec[3], Glove[4] and Huang[5]). All English word embeddings were downloaded on the Internet. Chinese embeddings were trained by gensim[6]. Statistics of the embeddings are shown in Table 2. We use the first four English embeddings on English dataset, and use the last two Chinese embeddings on FUDAN dataset.

In the experiment, we use the above embedding to initialize lookup tables of the model. Words not presented in the embedding are randomly initialized in the range $[-0.25, 0.25]$. All lookup tables are fine-tuned during training.

3.3. **Experiment settings.** We preprocess the data as follows. The preprocessing methods of the English datasets are consistent with previous papers [11,12]. For Chinese corpus FUDAN, only the Chinese characters, comma, period and question mark of the corpus are retained. Then, we segment the text by open source Python word segmentation tool Jieba[7].

We employ two frequently-used regularization methods on last fully-connected layer of the model: dropout [15] and l_2 weight normalization. The same hyper-parameters are used in all experiments: dropout rate is 0.5, l_2 constraint is 3 and mini-batch size is 50. Because FUDAN and AG_NEWS do not have a validation set, we randomly select 10% of the training data as validation set. During the training, we record the model which has best performance on validation set as the final model. Training is done through stochastic gradient descent over shuffled mini-batches with the Adadelta update rule.

4. **Results and Discussion.** Experimental results of the model are shown in this section. The further contrast experiment is used to analyze the models ability of adaptively integrating multiple word embedding.

4.1. **Sentiment classification.** A variety of baselines are used in this experiment: LR/ SVM (Logistic regression and SVM classifier with n-gram features, implemented by scikit-learn[8]), Recursive-NN (Three recursive neural network based models, Recursive Auto-Encoder (RAE) [7], Matrix-Vector Recursive Neural Network (MV-RNN) [8] and Recursive Neural Tensor Networks (RNTN) [9]), CNN (Two convolution networks, CNN-rand/ CNN-static/ CNN-non-static/ CNN-multichannel proposed in paper [11] and DCNN [10]).

The experimental results of the sentiment classification are listed in Table 3. It shows that the proposed method outperforms the traditional methods (LR/SVM) and the recursive-based models and obtains a comparable result with CNNs baseline. Because CNN

[3]http://code.google.com/p/word2vec/
[4]http://nlp.stanford.edu/projects/glove/
[5]http://ai.stanford.edu/ ehhuang/
[6]https://github.com/RaRe-Technologies/gensim
[7]https://github.com/fxsjy/jieba
[8]https://github.com/scikit-learn/scikit-learn

Table 3. Accuracy of the sentiment classification

Model		SST1 (%)	SST2 (%)
1-LR		37.0	81.8
2-LR		39.5	82.8
3-LR		39.0	82.9
1-SVM		39.3	81.8
2-SVM		40.1	81.8
3-SVM		40.9	82.5
RAE [7]		43.2	82.4
MV-RNN [8]		44.4	82.9
RNTN [9]		45.7	85.4
DCNN [10]		48.5	86.8
CNN-rand [11]		45.0	82.7
CNN-static [11]		45.5	86.8
CNN-non-static [11]		48.0	87.2
CNN-multichannel [11]		47.4	88.1
Our (ALL)	sum	41.4	84.5
	max	42.4	83.4
	mean	47.5	86.5

models use complex convolution operation to capture the contextual information of words, it may generate text representation with more word location information. In order to reduce the computational complexity and speed up the training, our model uses simple vector operation to combine word vector matrix when constructing the text feature vector, which may lose some information of the words order. However, the proposed method can be easily extended. The combination operation in our model can be easily replaced by other more sophisticated methods such as convolution.

4.2. **News classification.** We compared our model with baselines SVM/LR, Labeled-LDA [1] and CNN-rand in FUDAN dataset. The setting of SVM/LR is the same with sentiment classification, CNN-rand is implemented by ourselves and follows the hyper-parameter setting [11]. In AG_NEWS dataset, we choose a series of state-of-the-art models to compare with our model, including char-CNN [12], VDCNN [13] and fastText [14].

The experimental results of the news classification are listed in Table 4 and Table 5. (1) Our model outperforms all baselines on FUDAN dataset. The classification accuracy of traditional methods (LR/SVM) is nearly 10% less than other methods. This is due

Table 4. Accuracy of the FUDAN

Model		FUDAN (%)
1-LR		83.33
2-LR		84.61
3-LR		79.22
1-SVM		80.61
2-SVM		81.10
3-SVM		80.59
Labeled-LDA [1]		90.80
CNN-rand		91.89
Our (ALL)	sum	91.14
	max	87.21
	mean	92.64

TABLE 5. Accuracy of the AG_NEWS

Model		AG (%)
Bow [12]		88.8
Ngrams [12]		92.0
Ngrams TFIDF [12]		92.4
char-CNN [12]		87.2
VDCNN [13]		91.3
fastText [14]		92.5
Our (ALL)	sum	91.4
	max	92.3
	mean	91.3

to the fact that the words distribution in Chinese text is very sparse, and that n-gram feature often faces more serious data sparsity problem. Due to the fact that our model does not have data sparse problem, it can be effectively adapted to the language which has sparse word distribution, such as Chinese. (2) In AG_NEWS dataset, our model achieves similar results to the state-of-the-art methods. Surprisingly, our methods only use a simple shallow model to obtain better results than some recently proposed deep neural network models, such as char-CNN and VDCNN. This further verifies that our model can generate high quality text representation by integration of multiple pre-trained word embedding with rich semantic information.

4.3. **Ability of multiembedding integration.** In order to verify that the proposed model has the ability to improve the classification performance by combining multiple word embedding, some further experiments were conducted as follows. (1) We use single embedding to generate the text feature vectors and use this vector as final text representation to classify. (2) We repeat the experiment in Sections 4.1 and 4.2 without Highway network.

The experimental results are listed in Table 6; it shows that the classification accuracy is worse than the original model whether using single embedding or without Highway network. It proves that our model leverages the different semantic information brought by multiple word embedding when constructing the text representation. Because the contribution of different embeddings to the classification task is not the same, those feature vectors generated by unimportant embedding may cause a lot of noises in the final text representation. After joining the Highway network, our model can automatically identify the unimportant word embedding and guarantee the classification accuracy by adjusting text feature vectors adaptively.

5. **Conclusions.** This paper proposes a novel self-adaptive classification method based on multiple word embedding and neural network. The method can generate high quality distributed text representation by integration of multiple pre-trained word embedding with rich semantic information and improve classification performance obviously. A self-adaptive mechanism implemented by Highway network is adopted in the proposed method. It can adaptively distinguish the contribution of each word embedding in classification task and adjust the influence of different embedding for the final text representation.

We will explore more in future as follows. (1) In this paper, we use a Highway network to automatically adjust the text feature vectors generated by multiple word embedding. Recently, more and more researchers have begun to focus on attention-based neural networks. We will explore how to introduce the attention mechanism to distinguish the importance of each word embedding for text classification. (2) The proposed method uses simple vector operation to construct text feature vectors, and it may lose some information of the word context. We will explore better ways to construct text feature vectors.

TABLE 6. Results of the further experiment

Model		SST1 (%)	SST2 (%)	AG (%)
Our (Word2vec_EH)	sum	45.15	85.33	92.01
	max	44.88	83.90	91.86
	mean	45.83	83.52	91.60
Our (Glove_300)	sum	41.35	84.18	91.86
	max	44.97	84.18	90.23
	mean	44.47	84.56	91.57
Our (Glove_50)	sum	42.21	82.15	91.19
	max	38.14	79.73	88.92
	mean	29.59	71.77	78.53
Our (Huang)	sum	40.85	81.82	90.64
	max	36.83	80.01	88.42
	mean	31.08	74.90	81.14
Our (ALL-HW)	sum	43.30	83.63	91.40
	max	45.06	83.63	91.38
	mean	45.38	84.45	91.02
Our (ALL)	sum	43.62	84.51	91.43
	max	45.44	83.41	92.31
	mean	47.51	86.56	91.39

ALL-HW: Use all word embeddings but remove highway layer.

Acknowledgment. This paper was sponsored by Jilin Provincial Science and Technology Department of China (Grant No. 20170204002GX), Changchun Science and Technology Bureau of China (Grant No. 14KP009), Jilin Province Development and Reform Commission of China (Grant No. 2014Y056), and the National Science Foundation for Young Scholars of China (Grant No. 11501095). We would like to thank the organizations for their support.

REFERENCES

[1] W. Li, L. Sun and D. Zhang, Text classification based on labeled-LDA model, *Chinese Journal of Computers*, vol.31, no.4, pp.620-627, 2008.

[2] F. Yang, X. Zhou, D. Wu, X. Yang and T. Sun, A new method of Chinese short text classification based on the domain ontology, *ICIC Express Letters*, vol.6, no.6, pp.1399-1404, 2012.

[3] D. Wu, Y. Wang and F. Yang, Improvement of term weighting algorithm in text classification, *ICIC Express Letters*, vol.6, no.5, pp.1347-1352, 2012.

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

[5] Y. Chen, B. Perozzi, R. Al-Rfou and S. Skiena, The expressive power of word embeddings, *arXiv:1301.3226*, 2013.

[6] F. Hill, K. Cho, S. Jean, C. Devin and Y. Bengio, Not all neural embeddings are born equal, *arXiv:1410.0718*, 2014.

[7] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng and C. D. Manning, Semi-supervised recursive autoencoders for predicting sentiment distributions, *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pp.151-161, 2011.

[8] R. Socher, B. Huval, C. D. Manning and A. Y. Ng, Semantic compositionality through recursive matrix-vector spaces, *Proc. of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp.1201-1211, 2012.

[9] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng and C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, *Proc. of the Conference on Empirical Methods in Natural Language Processing*, p.1642, 2013.

[10] N. Kalchbrenner, E. Grefenstette and P. Blunsom, A convolutional neural network for modelling sentences, *arXiv:1404.2188*, 2014.

[11] Y. Kim, Convolutional neural networks for sentence classification, *arXiv:1408.5882*, 2014.

[12] X. Zhang, J. Zhao and L. Yann, Character-level convolutional networks for text classification, *Advances in Neural Information Processing Systems*, pp.649-657, 2015.
[13] A. Conneau, H. Schwenk, L. Barrault and L. Yann, Very deep convolutional networks for natural language processing, *arXiv:1606.01781*, 2016.
[14] A. Joulin, E. Grave, P. Bojanowski and T. Mikolov, Bag of tricks for efficient text classification, *arXiv:1607.01759*, 2016.
[15] R. K. Srivastava, K. Greff and J. Schmidhuber, Highway networks, *arXiv:1505.00387*, 2015.
[16] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research*, vol.15, no.1, pp.1929-1958, 2014.