MI&T Lab at SemEval-2017 task 4: An Integrated Training Method of Word Vector for Sentiment Classification

Jingjing Zhao, Yan Yang, Bing Xu
Laboratory of Machine Intelligence and Translation, Harbin Institute of Technology
School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
zhaojingjing@hit.edu.cn, yangyan@hit.edu.cn, hitxb@hit.edu.cn

Abstract

A CNN method for sentiment classification task in Task 4A of SemEval 2017 is presented. To solve the problem of word2vec training word vector slowly, a method of training word vector by integrating word2vec and Convolutional Neural Network (CNN) is proposed. This training method not only improves the training speed of word2vec, but also makes the word vector more effective for the target task. Furthermore, the word2vec adopts a full connection between the input layer and the projection layer of the Continuous Bag-of-Words (CBOW) for acquiring the semantic information of the original sentence.

1 Introduction

The polarity of a Twitter message is classified into positive, negative and neutral in Twitter sentiment analysis. However, the difficulty of sentiment analysis greatly increases due to the ambiguity and the rhetorical of natural language (Liu, 2012). In recent years, the deep learning model has shown great potential in the task of sentiment classification (Socher et al., 2011; Poria et al., 2015; Socher et al., 2013). For short text data such as Twitter, Convolutional Neural Network (CNN) model (Kim, 2014; Dos Santos and Gatti, 2014; Chen et al., 2016) is the most widely and successfully used, and in the SemEval 2016-task4A competition, the system ranked first also uses CNN model (Deriu et al., 2016). So CNN model is used to complete the task in our system. The task 4A of SemEval 2017[1] is a polarity classification task which requires participated systems to classify a given Twitter message into positive, negative or neutral (Rosenthal et al., 2017).

The system integrates the word2vec and CNN to train the labeled data, generating the word vector of each word in the data. This method can improve the training speed of word vector. In order to preserve the more semantic information of the original sentence effectively, the word2vec is fully connected between the input layer and the projection layer of the Continuous Bag-of-Words (CBOW).

2 System description

2.1 Word vector representation method

Word2vec can represent every word that appears in a large number of training texts as a lower dimension vector (usually 50-100 dimensions). (Mikolov et al., 2013b; Rong, 2014; Mikolov et al., 2013a) have a detail description of word2vec. Word2vec in our system uses Continuous Bag-of-Words (CBOW) model, and the structure is shown in Figure 1, in which w_i is a word, and the sequence $\langle w_{(i-c)}, ..., w_{(i-1)}, w_{(i+1)}, ..., w_{(i+c)} \rangle$ represents the context of the word w_i, and c is the window size. The word-vector length of the word w_i is d. The traditional word2vec adds $2c$ words' word vector on the input layer to the projection layer. (Mikolov et al., 2013b) has a detail description of this method, which has been exactly used in Google Word2vec released in 2013. However, in the sentiment analysis task, whether there is a negative word before the sentiment word will influence the identification of polarity (Liu, 2012). So in order to preserve more emotional semantic information, the input layer and the projection layer of CBOW are fully connected in this system.

The procedure of integrating word2vec and CNN to train the words vector follows four

[1] http://alt.qcri.org/semeval2017/task4/

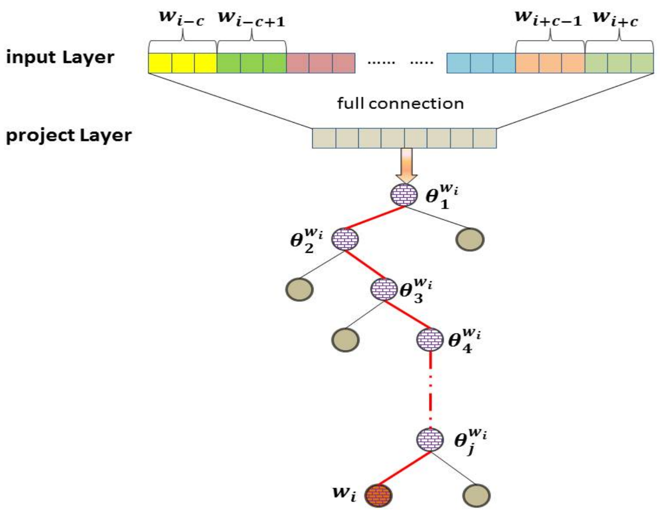

Figure 1: The structure of word2vec in the system.

steps: *(i)* initialization and pre-training: initialize word2vec and CNN parameters, and pre-train word2vec a certain number of iterations; *(ii)* CNN training: input the latest words vector from word2vec to CNN; *(iii)* word2vec training: input the latest words vector from CNN to word2vec; *(iv)* alternate the *(ii)* and *(iii)* steps until the training phase converges. CNN can help word2vec extract more effective text features. Experiments show that the model obtained by integrating CNN and word2vec performs better when the data is sufficient compared to adopting them separately.

2.2 Deep learning model

The system proposes CNN model to predict the sentiment polarity of a Twitter text. The CNN structure diagram is shown in Figure 2.

Word vector sequence: Each word is represented as a d-dimensional word vector, a sentence or a Twitter text containing n words can be expressed as n d-dimensional vectors, which are concatenated together into a matrix $X \in \mathbb{R}^{d\times n}$ form which represents a sentence or a Twitter text. Each row of matrix X is treated as a new vector, thus d n-dimensional vectors are obtained and concatenated as input to the CNN network.

Convolutional layer: The convolution layer uses full convolution operation. Let $F_i^l \in \mathbb{R}^{M_1}$ represents ith feature map at lth layer[2], and $m_{j,i}^l \in \mathbb{R}^{M_2}$ represents the convolutional kernel of the jth convolutional result C_j^{l+1} at $l+1$th layer. So the jth convolutional result C_j^{l+1} at $l+1$th layer is the result of convolution operation between each feature map at lth layer and convolutional kernel $m_{j,i}^l$, i.e.,

$$C_j^{l+1} = \sum_j m_{j,i}^l * F_i^l \tag{1}$$

k-max pooling: After the convolution operation, the max-pooling operation is performed which preserves the largest value in each convolution result. The system uses k-max pooling, which preserves the largest k values instead. For example:

$$(1,6,3,8) \xrightarrow{2-max} (6,8)$$

Where k is a parameter that needs to be set manually.

full-connection Layer: The full connection layer receives the output of the last layer of CNN and fully connects itself to the output layer, i.e. $W*x+b$ operation, where W and b can be trained during the network training phase.

output Layer: The output layer uses the Softmax operation and outputs the probability that the input sentence or Twitter text belongs to each class, and the class with the maximum probability is the predicted class judged by the system.

2.3 Combination prediction

Because of the insufficiency of training data and the great quantity bias in different classes' training data, the trained CNN can't work so well. So our system adds Support Vector Machine (SVM) (Suykens, 2001) model to predict jointly with

[2]Here, a layer refers to one convolutional and one pooling layer.

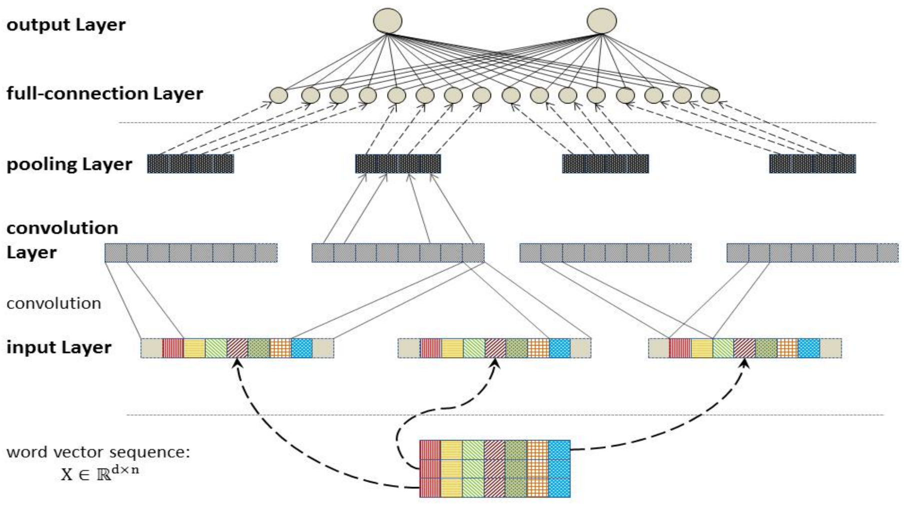

Figure 2: The structure of CNN in the system.

CNN. The steps of combination prediction are shown in Figure 3.

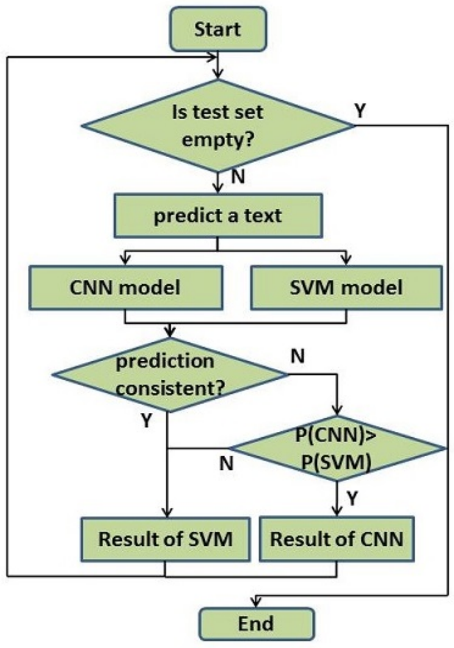

Figure 3: The steps of combination prediction.

3 Experimental datasets and model parameters

3.1 Experimental datasets

The datasets of the experiment is provided by SemEval 2017, and the specific datasets used are shown in Table 1.

Dataset	positive	negative	neutral	Total
train	18123	14354	21748	54225
dev	1000	500	1500	3000
test	2375	3972	5937	12284

Table 1: Datasets of the experiment.

3.2 Model parameters

The parameters set of word2vec and CNN in our system are shown in Table 2.

word2vec parameters	
Window size of context c	3
Number of hidden layer	1
Dimensionality of hidden layer	[50]
Convolutional Neural Network parameters	
Number of Convolutional layer	1
Number of max-pooling layer	1
Convolutional kernel size	7
Number of feature map at Convolutional	100
k-max pooling parameter *k*	2

Table 2: Parameters used in the word2vec and CNN.

4 Experimental results and analysis

4.1 Evaluation method

The measure metric of the Evaluation is average macro recall (Rosenthal et al., 2017). The formula is as follows:

$$\rho^{PUN} = \frac{\rho^P + \rho^U + \rho^N}{3} \tag{2}$$

Here, ρ^P, ρ^U and ρ^N denote recall for the positive class, neutral class and negative class.

The other two measure metrics are the average macro F_1 and the average macro precision:

$$F_1^{PUN} = \frac{F_1^P + F_1^U + F_1^N}{3} \tag{3}$$

$$P^{PUN} = \frac{P^P + P^U + P^N}{3} \tag{4}$$

Here, F_1^P, F_1^U and F_1^N denote F_1 for the positive class, neutral class and negative class; P^P, P^U and P^N denote precision for the positive class, neutral class and negative class.

4.2 Analysis of experimental results

Table 3 lists the average macro recall for each model on the development dataset. From the table 3, the effect of word2vec+CNN model is better than SVM, and word2vec + CNN + SVM is the best of the three models, so the best results on the test set are submitted.

System	**SVM**	**CNN**	**SVM+CNN**
ρ^{PUN}	0.589	0.601	0.653

Table 3: The results of different models on the development dataset.

Table 4 shows the details of our system's result in comparison with the three top ranked systems' results. It can be seen from the table that our result's ρ^N is not good, but ρ^U is better than the top three systems. The decrease of experimental results is from the quantity bias in training data of different classes.

For deep learning models, a lot of training data are required. Due to the lack of Twitter texts, word2vec training is not sufficient and do not generate effective words vector representation. In the future, semi-supervisory mechanisms will be considered to expand the number of training data.

In the future, we can improve the system's performance from following points: *(i)* to expand the amount of training data; *(ii)* to improve the type of combination: the results can be combined with multiple CNN systems to predict; *(iii)* to add more emotional semantic features.

5 Conclusion

This paper presentes a method of training word vector by integrating word2vec with CNN and using the trained CNN to complete the Twitter sentiment analysis task. In the future work, we hope to continue to improve system's performance in multiple ways, such as trying to modify some parameters or improve the type of classifiers' combination, adding some sentiment features.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61402134).

References

Peng Chen, Bing Xu, Muyun Yang, and Sheng Li. 2016. Clause sentiment identification based on convolutional neural network with context embedding. In *Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016 12th International Conference on*. IEEE, pages 1532–1538.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurelien Lucchi, Valeria De Luca, and Martin Jaggi. 2016. Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. *Proceedings of SemEval* pages 1124–1128.

Cícero Nogueira Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*. pages 69–78.

System	Positive			Negative			Neutral			Score
	P	ρ	F_1	P	ρ	F_1	P	ρ	F_1	ρ^{PUN}
DataStories(1)	0.6259	0.7023	0.6619	0.5929	**0.8291**	0.6914	0.7471	**0.5115**	0.6073	0.681
BB twtr(1)	0.6851	0.6522	0.6682	0.5848	**0.8776**	0.7019	0.7518	**0.5144**	0.6109	0.681
LIA (3)	0.6480	0.6518	0.6499	0.6082	**0.8170**	0.6973	0.7289	**0.5599**	0.6333	0.676
MI&T Lab	0.4780	0.5171	0.4968	0.5496	**0.5451**	0.5473	0.6066	**0.5902**	0.5983	**0.551**

Table 4: The details of our result and the three top ranked results.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5(1):1–167.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Soujanya Poria, Erik Cambria, and Alexander F Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *EMNLP*. pages 2539–2544.

Xin Rong. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738* .

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Vancouver, Canada, SemEval '17.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 151–161.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. Citeseer, volume 1631, page 1642.

Johan AK Suykens. 2001. Nonlinear modelling and support vector machines. In *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*. IEEE, volume 1, pages 287–294.