East Asian Math. J.
Vol. 37 (2021), No. 5, pp. 599–612
http://dx.doi.org/10.7858/eamj.2021.039

RESEARCH ON SENTIMENT ANALYSIS METHOD BASED ON WEIBO COMMENTS

Zhong-Shi Li, Lin He, Wei-Jie Guo, and Zhe-Zhi Jin*

Abstract. In China, Weibo is one of the social platforms with more users. It has the characteristics of fast information transmission and wide coverage. People can comment on a certain event on Weibo to express their emotions and attitudes. Judging the emotional tendency of users' comments is not only beneficial to the monitoring of the management department, but also has very high application value for rumor suppression, public opinion guidance, and marketing. This paper proposes a two-input Adaboost model based on TextCNN and BiLSTM. Use the TextCNN model that can perform local feature extraction and the BiLSTM model that can perform global feature extraction to process comment data in parallel. Finally, the classification results of the two models are fused through the improved Adaboost algorithm to improve the accuracy of text classification.

1. Introduction

At present, there are three methods for sentiment analysis, namely, the method based on the sentiment dictionary, the method based on traditional machine learning, and the method based on deep learning.

Sentiment analysis method based on sentiment dictionary. It is an unsupervised method that requires human participation. It can classify the text by calculating the appearance of emotional words in the dictionary in the analysis text. Huettner et al[1] used fuzzy logic to study document-level sentiment analysis. They artificially constructed a dictionary covering 4000 sentiment words and divided them into categories; Tang et al[2] used the method of representation learning to learn the emotional learning of the text and constructed the emotional dictionary of Twitter; Keyu et al[3] based on the extended sentiment dictionary, carried out sentiment classification on the hotel field evaluation; Kouloumpis et al[4] used four characteristics to classify Weibo sentiment. However, the emotion classification based on the emotion dictionary also has some

Received June 2, 2021; Revised September 24, 2021; Accepted September 29, 2021.
2010 *Mathematics Subject Classification.* 11A11.
Key words and phrases. Emotion Analysis; TextCNN Model; BiLSTM Model; Adaboost Algorithm.
*Corresponding author.

(pISSN 1226-6973, eISSN 2287-2833)

shortcomings. For example, with the rapid development of the Internet, some words are not the meanings in people's cognition, but suggest other meanings. For this kind of new emotion words, There will be a certain error in the emotion classification. In this regard, sentiment analysis methods based on machine learning have gradually developed.

Based on traditional machine learning methods, it also requires human participation. After artificially constructing features, a machine learning algorithm is used to design a classifier to find the optimal solution of the objective function based on the text features. In fact, this is similar to the optimization problem in mathematics. Turney et al[5] proposed a simple unsupervised algorithm, which first uses a part-of-speech tagger to identify adjectives or adverb phrases in the text, and then classifies emotions based on the average value of each extracted phrase; In the same year, Pang et al[6], based on Turny's research, proposed for the first time the use of machine learning algorithms for emotion classification, using naive Bayes, maximum entropy models, and support vector machines for three machine learning methods.

In recent years, deep learning methods have gradually been applied to the research of sentiment analysis. Kim et al[7] used Convolutional Neural Networks (CNN) to conduct experimental comparisons by changing the size of the convolution kernel and using different pooling methods. Wang et al[8] used long short-term memory network (LSTM) to conduct sentiment analysis research based on Twitter's comment data; Pal et al[9] adopted a two-way LSTM to propagate the data forward and backward, and improved the classification accuracy by superimposing the LSTM layer. In order to be able to obtain more information we want from a large amount of data, the technology we use is gradually improving. Compared with methods based on machine learning, in terms of sentiment analysis, deep learning methods are slightly better.

2. Related theories

2.1. TextCNN model

The core idea of convolutional neural network is to capture local features. For text classification, local features are sliding windows composed of several words. Compared with traditional text classification algorithms, CNN can efficiently extract features in text and improve the efficiency of text classification.

The model structure of the TextCNN network is shown in Figure 1.1. The model is composed of input layer, convolutional layer, pooling layer and fully connected layer.

(1) The first layer is the input layer. It is an $n*k$ matrix, Where n represents the number of words in the sentence, and k represents the dimension of each word vector. In order to ensure that the length of the vector is consistent with the original sentence, padding is performed on the original sentence. The word vector can be extracted from the trained corpus, or it can be trained by the

network. The word vectors of the trained corpus can get more prior knowledge, and the word vectors without training are more in line with the current task.

(2) The second layer is a convolutional layer. Unlike images, text is a vector matrix composed of a series of word vectors, and the width of the convolution kernel is the same as the width of the word matrix. The convolution kernel will only move in the vertical direction of the word vector. In this way, it can be ensured that each sliding position of the convolution kernel is a complete word vector. The rows of the word matrix represent discrete symbols, which ensures the rationality of words as the smallest granularity in the language.

(3) The third layer is the pooling layer. Using 1-max pooling for pooling means that the largest feature value is selected from the feature vector generated by each sliding window, and these selected features are finally spliced to form a vector. In addition, k-max pooling or average pooling can also be used to compress features.

(4) The last layer is a fully connected layer. After the previous network operation, we can get the vector representation of the text, the vector is input into the fully connected layer for classification, and the softmax activation function is used to output the probability of classification.

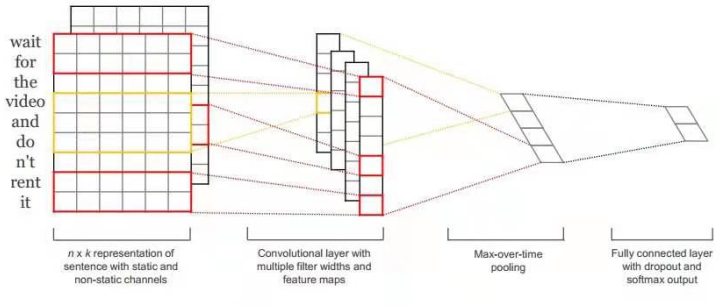

FIGURE 1. TextCNN model structure

2.2. BiLSTM model

When LSTM is processing text, its advantage is that it can capture the dependency relationship between words at a longer distance. By training the LSTM, the model can learn to remember and forget the information. But in the process of training the LSTM to model the sentence, the LSTM algorithm encodes the sentence from front to back according to the order of the sentence, but it cannot obtain the information from the back to the front of the sentence.

In view of the shortcomings of LSTM, Cross J et al[10] proposed the BiLSTM (Bidirectional Long Short-Term Memory) algorithm, which is composed of two LSTMs, namely the forward LSTM algorithm and the backward LSTM algorithm. The BiLSTM network process is shown in Figure 2.

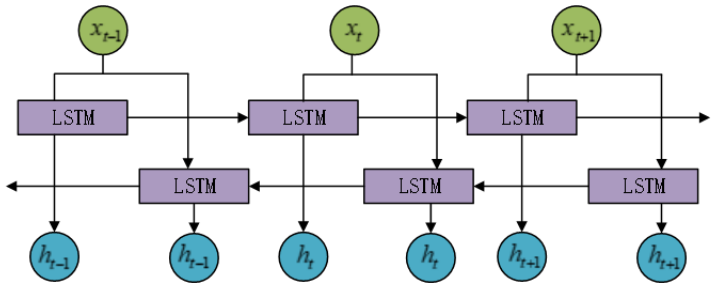

FIGURE 2. BiLSTM model structure

As can be seen from Figure 2, BiLSTM is composed of an input layer, a forward propagation LSTM layer, a backward propagation LSTM layer and an output layer. The forward LSTM encodes the sentence from front to back to obtain the logical relationship between positive words, and the directional LSTM encodes the sentence from back to front to obtain the logical relationship between reverse words.

The mathematical expression of forward LSTM layer is shown in (1).

$$\begin{cases} \overrightarrow{i_t} = \sigma\left(\overrightarrow{W}_t \overrightarrow{x}_t + \overrightarrow{V}_i \overrightarrow{h}_{t-1} + \overrightarrow{b}_i\right) \\ \overrightarrow{f}_t = \sigma\left(\overrightarrow{W}_f \overrightarrow{x}_t + \overrightarrow{V}_f \overrightarrow{h}_{t-1} + \overrightarrow{b_f}\right) \\ \overrightarrow{o}_t = \sigma\left(\overrightarrow{W}_f \overrightarrow{x}_t + \overrightarrow{V}_o \overrightarrow{h}_{t-1} + \overrightarrow{b}_o\right) \\ \overrightarrow{c}_t = \overrightarrow{f}_t \cdot \overrightarrow{c}_{t-1} + \overrightarrow{i}_t \cdot \tanh\left(\overrightarrow{W}_c \overrightarrow{x}_t + \overrightarrow{V}_c \overrightarrow{h}_{t-1} + \overrightarrow{b}_c\right) \\ \overrightarrow{h}_t = \overrightarrow{o}_t \cdot \tanh\left(\overrightarrow{c}_t\right) \end{cases} \tag{1}$$

Where, $\overrightarrow{x}_t$ represents the input at the momentt, and $\overrightarrow{h}_{t-1}$ represents the hidden layer state value at the momentt-1;

$\overrightarrow{V}_i, \overrightarrow{V}_f, \overrightarrow{V}_0$ and $\overrightarrow{V}_c$ represent the weight coefficients of $\overrightarrow{h}_{t-1}$ during the forgetting gate, input gate, output gate, and feature extraction, respectively;

$\overrightarrow{W}_i, \overrightarrow{W}_f, \overrightarrow{W}_0$ and $\overrightarrow{W}_c$ represent the weight coefficients of $\overrightarrow{x}_t$ during the forgetting gate, input gate, output gate, and feature extraction respectively;

$\overrightarrow{b}_i$, $\overrightarrow{b}_f$, $\overrightarrow{b}_0$ and $\overrightarrow{b}_c$ represent the offset values during the forgetting gate, input gate, output gate, and feature extraction, respectively;

tanh represents the positive tangent hyperbolic function and σ respectively represents the activation function Sigmoid.

The mathematical expression of the backward LSTM layer is as follows:

$$\begin{cases} \overleftarrow{i_t} = \sigma\left(\overleftarrow{W}_t \overleftarrow{x}_t + \overleftarrow{V}_i \overleftarrow{h}_{t-1} + \overleftarrow{b_i}\right) \\ \overleftarrow{f_t} = \sigma\left(\overleftarrow{W}_f \overleftarrow{x}_t + \overleftarrow{V}_f \overleftarrow{h}_{t-1} + \overleftarrow{b_f}\right) \\ \overleftarrow{o_t} = \sigma\left(\overrightarrow{W}_f \overleftarrow{x}_t + \overleftarrow{V}_o \overleftarrow{h}_{t-1} + \overleftarrow{b}_o\right) \\ \overleftarrow{c_t} = \overleftarrow{f}_t \cdot \overleftarrow{c}_{t-1} + \overleftarrow{i}_t \cdot \tanh\left(\overleftarrow{W}_c \overleftarrow{x_t} + \overleftarrow{V}_c \overleftarrow{h}_{t-1} + \overleftarrow{b_c}\right) \\ \overleftarrow{h}_t = \overleftarrow{o}_t \cdot \tanh\left(\overleftarrow{c_t}\right) \end{cases} \tag{2}$$

$\overrightarrow{h}_i$ and $\overleftarrow{h}_i$ respectively represent the output vectors of the forward LSTM and the reverse LSTM at time t. Therefore, the output layer vector of BiLSTM is shown in formula (3):

$$h_t = \left[\overrightarrow{h}_t \oplus \overleftarrow{h}_t\right] \tag{3}$$

Wherein, $\oplus$ represents the output format of BiLSTM, including sum, concat, ave, mul, and so on.

2.3. Adaboost model

The Adaboost algorithm is an improved ensemble algorithm based on Boosting. Its core idea is to train different classifiers (weak classifiers) for the same training set, and then combine several weak classifiers by linear weighted combination to form the final composition a strong classifier.

In the process of the Adaboost algorithm, each data sample is initialized first, and the weight of each sample conforms to a uniform distribution, and is used to calculate the next algorithm iteration. The algorithm solves some difficult-to-classify samples by increasing the weight of the samples that are misclassified in the data and reducing the weight of the samples that are correctly classified. By increasing the weight of misclassified samples, it will be paid more attention to in the subsequent calculations. By analogy, the weak learners are integrated to construct a strong learner. The specific algorithm flow of Adaboost is as follows:

(1) Training sample weight: First, assign weights equally to N training samples, that is, get the weight of each training sample as $\frac{1}{N}$, and denote the weight set as D_1, and its sum is 1, which can be regarded as a probability distribution. As shown in the formula (4):

$$D_1 = \left(\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}\right) \tag{4}$$

(2) Calculate the classification error: Calculate the classification error rate of the classifier on the training data set. As shown in the formula (5):

$$\varepsilon_t = \sum_{i=1}^{N} D_i^t \left[h_t\left(x_i\right) \neq y_i\right] \tag{5}$$

Among them, ε_t represents the classifier error; D_i^t represents the weight of the i-th sample at the t-th iteration; $h_t(x_i)$ represents the classification category of x_i the sample at the t-th iteration, and y_i represents the true label of the x_i sample.

(3) Calculate the weight of the classifier: determine the proportion coefficient of the weak classifier according to the classification error. The smaller the error is, the larger the coefficient is, and the larger the decision proportion in the final classifier is. The larger the error is, the smaller the coefficient is, and the smaller the decision proportion in the final classifier is. The calculation is shown in formula (6):

$$\alpha(t) = \frac{1}{2}\log\frac{1-\varepsilon^t}{\varepsilon^t} \tag{6}$$

(4) Update sample weight: adjust the sample weight according to the classification error, so that the weight of the classifier with small classification error in the previous round is reduced, and the weight of the classifier with large classification error is increased, which can be regarded as a new probability distribution. The formula is shown in (7):

$$D_i^{t+1} = D_i^t \cdot \exp\left(\alpha_t \cdot \left[h_t(x_i) \neq y_i\right]\right) \tag{7}$$

(5) Output strong classifier: Construct a linear combination of basic classifiers to obtain the final strong classifier. The calculation is shown in the formula (8):

$$H(x) = \sum_{t=1}^{T} \alpha(t)\left[h(x_i) = y_i\right] \tag{8}$$

There are two kinds of weights in the Adaboost algorithm, one is the weight of the data, and the other is the weight of the weak classifier. Among them, the weight of the data is mainly used for the weak classifier to find the decision point with the smallest classification error, and then use this minimum error to calculate the decision weight of the weak classifier in the final strong classifier.

3. Model Design

3.1. Algorithm improvements

In this paper, the Weibo comment text fine-grained emotion analysis model is designed: the TextCNN model which can extract local features and the BiLSTM model which can extract global features in parallel to classify the data in the text classification layer, and finally the results are fused by the Adaboost algorithm. The Adaboost algorithm requires the base classifier to be homogeneous classifier, and in the Adaboost algorithm based on two inputs, local and global features are trained in parallel, so the classifiers that classify local features and classifiers that classify global features can be homogeneous classifiers, or heterogeneous classifiers, and the classifiers trained in this paper are heterogeneous classifiers. Compared with the original algorithm, the improved Adaboost algorithm has

the following advantages: (1) the input signal changes from the original single input to multiple inputs; (2) the input of the model is in parallel input form, and the classifiers for each branch can be homogeneous classifiers or heterogeneous classifiers. The training process in details is as follows: First, based on the classification accuracy of the TextCNN model and the BiLSTM model, the weights of the two classifiers are calculated: the calculation is shown in the formula (9) to (10):

$$\alpha_1^{(t)} = \frac{1}{2}\log\frac{1-\varepsilon_1^{(t)}}{\varepsilon_1^{(t)}} \tag{9}$$

$$\alpha_2^{(t)} = \frac{1}{2}\log\frac{1-\varepsilon_2^{(t)}}{\varepsilon_2^{(t)}} \tag{10}$$

According to the formula above, trepresents the number of iterations, $\alpha_1^{(t)}$ is the TextCNN classifier's weight after the t-th iteration, $\alpha_2^{(t)}$ represents the BiLSTM classifier's weight after the t-th iteration, $\varepsilon_1^{(t)}$ is TextCNN classifier's classification error rate, and $\varepsilon_2^{(t)}$ is the BiLSTM classifier's classification error rate. After each iteration, the weights of the two classifications are adjusted according to the training results, that is, the weight of the misclassified sample is increased and the weight of the correct sample is reduced. The formula (11) is shown as follows:

$$D_i^{t+1} = D_i^t \cdot \exp\left(\sum_{j=1}^{2}\alpha_j^{(t)} \cdot \max\left(0, \frac{1}{2}\operatorname{sgn}\left(z_i\right)\right)\right) \tag{11}$$

In detail, $\operatorname{sgn}(z_i)$ is the symbolic function, and $\operatorname{sgn}(z_i) = |h_t(x_i) - y_i| \cdot h_t(x_i) \neq y_i$. Finally, based on the weight of the classifier obtained for each iteration, the strong classifier is superimposed:

$$H(x_i) = \sum_{t=1}^{T}\sum_{j=1}^{2}\left(\alpha_j^{(t)} C_j^{(t)}(x_i)\right) \tag{12}$$

The iterative process of entering the Adaboost model is shown below. Firstly, the text word vector features are entered into TextCNN and BiLSTM for independent training, and the two algorithms are aiming at extracting the local and global characteristics of the word vector, respectively. And then classify them. At the end of the iteration, you get the classification errors and classifier weights of the TextCNN classifier and BiLSTM classifier, which feeds back to each other according to the classification results of the two classifiers, and adjusts the sample of the misclassification. Next, the next iteration ends when the number of iterations reaches the upper limit or when the classification error is less than the threshold. Finally, the final strong classifier is formed. The two-input Adaboost model not only changes the input method, but also further updates the weight of the sample by adjusting the feedback from the two classification results before outputting the strong classifier.

The structure of the two-input Adaboost model is shown in Figure 3.

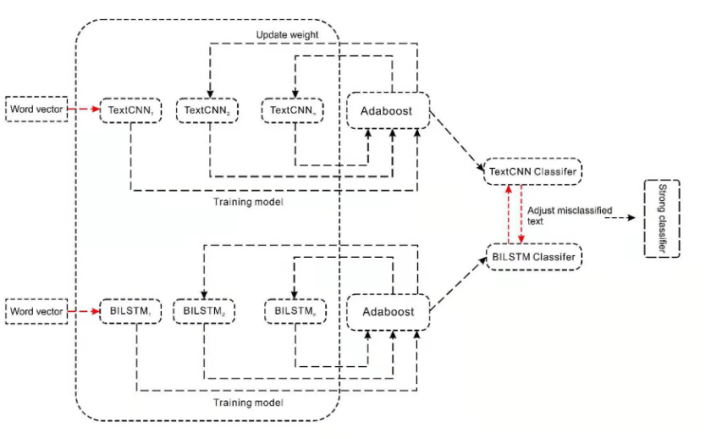

FIGURE 3. Two-input Adaboost model structure

3.2. Evaluation indicators

In order to fully analyze the validity of the text classification model, this paper analyzes the effect of the text classification model from indicators such as *accuracy*, *F1*, and *recall*, where accuracy represents the proportion of the correct sample in the total forecast sample; Precision represents the proportion of correctly predicted positive samples in the total forecast sample; *Recall* represents the recall rate, indicating that the correctly predicted number of positive samples represents the proportion of real positive samples; *F1* is the reconciling average of precision and recall. The definitions for TP, FP, and FN are shown in Table 1:

TABLE 1. Parameter interpretation

	Predictive Label	
The actual value (Actual Label)	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Formula (13) to (16) represents *accuracy*, *precision*, *recall* rate, and how F1 values are calculated.

$$accuracy = \frac{TP + TN}{total} \tag{13}$$

$$precision = \frac{TP}{TP + FP} \tag{14}$$

$$recall = \frac{TP}{TP + FN} \tag{15}$$

$$F1 = \frac{2 * P*R}{P + R} \tag{16}$$

4. Experiments and results analysis

4.1. The source of the experimental data

In the experiment, the data set was selected to crawl on the Sina Weibo platform. There were about 50,000 comments on user attitudes towards consumption during June in 2019 to December in 2020. In the original data without labels, randomly extract part of the data for pre-processing, manual labeling and other work. Manual labeling uses 0, 1, 2 to representing neutral, negative, positive respectively. The final 10,000 pieces of data with manual labels are divided into training sets, validation sets, and test sets in a scale of 8:1:1, as shown in Table 2.

Table 2. Dataset

Data set	Training set	The validation set	The test set
Quantity	8000	1000	1000

4.2. Experimental environment configuration

This lab environment configuration is shown in Table 3:

Table 3. Environment configuration instructions

The configuration name	illustrate
Memory	16G
Processor	Inter Xeon CPU E3-1230 v3
Operating system	Windows 10
Develop tools	PyCharm
Simulation platform	Python

4.3. Comparative experimental analysis

The experimental results of the text classification models: Bert model, GRU model, LSTM model, RNN model and text classification model are compared. The model proposed in this paper is based on the text classification of the two-input Adaboost model, which can be combined with the TextCNN algorithm and the BiLSTM algorithm by improving the Adaboost algorithm, and it is iterated by using 5 TextCNN algorithms and 5 BiLSTM algorithms respectively in the Adaboost model, and finally outputs a strong text classification model.

The validity of this model is proved by analyzing the classification results of different models by three evaluation indicators–*accuracy*, *F1* and *recall*.

Table 4 represents the accuracies of several classification models.

TABLE 4. The accuracy of the text classification of the model (%)

Model	Bert	GRU	LSTM	RNN	Our models
Accuracy	91.07	90.67	91.03	89.73	92.01

As can be seen from Table 4, the model of this paper has achieved the best results in five text classification models, with an accuracy rate of 92.01%, the RNN model has the lowest accuracy of 89.73%, and the accuracy of this model is 2.28%, higher than that of the RNN model. The accuracy of the Bert model is 91.07%, and the model of this article is superior to the classification accuracy of the Bert model. The LSTM model has an accuracy of 91.03%, which is second to the target model effect and the Bert model effectin this article. Behind the LSTM model is the GRU model, which achieves 90.67% accuracy. By analyzing the classification accuracy of the five models, the validity of the model is proved to be superior to the other four models.

Figure 4 represents the accuracy transformation trend of the five models in the text classification that iterate 20 times.

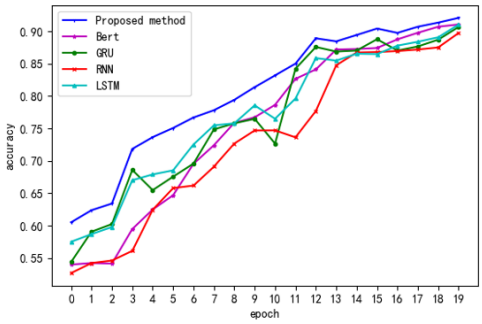

FIGURE 4. Model iterative graph (accuracy)

As you can be seen from Figures 3.1 and Table 3.3, the classification accuracy of the model increases as the number of iterations increases. The proposed model (proposed method) increases with the number of iterations, the accuracy increases more smoothly, and there will be no large fluctuation, and when the number of iterations reaches a certain value, the speed of the increase in accuracy tends to stabilize. Of the remaining four classification models, the LSTM model grows steadily, with significant oscillations at 11 iterations, and the stability of the model can be demonstrated at other value locations. However, in

GRU, RNN, and Bert models, the classification accuracy of the model varies with the number of iterations, and the corresponding accuracy value fluctuates significantly, indicating that the stability of the model needs to be further improved, but overall, the accuracy of the five models increases with the number of iterations. As can be seen from Figure 4, the model proposed in this paper is superior to the rest ones in accuracy and stability.

Table 5 represents the recall rates for the Bert model, the GRU model, the LSTM model, the RNN model, and the model in this article.

TABLE 5. Text Classification Recall Rate (%) for models

Model	Bert	GRU	LSTM	RNN	Our models
Recall rate	90.64	90.21	90.24	89.34	91.14

Table 5 represents the model effect as an indicator of recall rates, respectively. The recall rates from high to low is that of Our models, Bert model, LSTM model, GRU model and RNN model. Among them, the recall rate of this proposed model is the highest, 91.14%. Compared with the model and the Bert model, the difference between the two is 0.5%, which proves that the model of this paper is excellent, but also shows that the Bertmodel can achieve some results. The RNN model has the lowest recall rate and it is 89.34%, a 1.8% difference from the model proposed in this paper. By analyzing the classification recall rate of five models, the validity of this model is proved to be superior to that of the other four models.

Figure 5 represents a recall rate change trend of 20 iterations for each of the five models in the text classification.

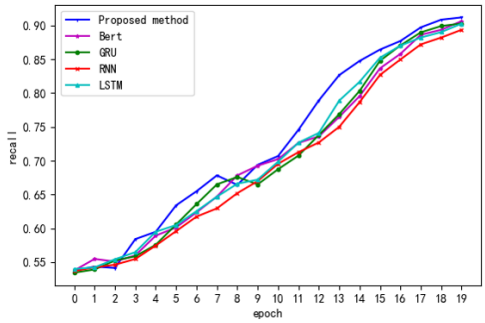

FIGURE 5. Model iterative graph (recall rate)

As can be seen from Figure 5 and Table 5, the recall rates of the five models are steadily increasing as the number of iterations increases. The figure shows that the proposed model in this article is generally superior to the recall rate of

other models. When the number of iterations is 8, the recall rate of this model fluctuates somewhat, but overall the model has strong stability. The RNN model exhibits the best stability, with an increase in recall rates as iterations increase, but the model has a poor recall rate, which is inferior to the effect of the remaining four models. The recall rate and stability of GRU model and LSTM model are quite good, but compared with the recall rate of this model, there is a certain gap in the recall rate in each iteration. Overall, the recall rates of the five models increases with the number of iterations, and the recall rate of the model presented in this paper is better than that of the other models in each iteration, but the stability of the model needs to be further improved.

Table 6 represents the F1 values of the Bert model, the GRU model, the LSTM model, the RNN model, and the model in this article.

TABLE 6. Text Classification F1 (%) of the model

Model	Bert	GRU	LSTM	RNN	Our models
F1 value	90.62	89.63	90.71	89.42	91.61

Table 6 represents the model effects of the five models under F1 values. The F1 values of the Bert model, the LSTM model and our model are all above 90%, of which the model works best and F1 value of it is 91.61%, the F1 value of LSTM model is 90.71%, and the Bert model has an F1 value of 90.62%. The difference between the F1 value of the LSTM model and the model presented in this article is 0.85%, and the Bert model is 0.9% from the F1 value of the model presentedinthis article, which shows that the model of this article is more effective than the other two models. The effect of the GRU model and the RNN model is comparable, with only a 0.21% difference between the two models, but the GRU model differs by 1.98% from the F1 value of the model in this article and 2.19% between the F1 value of the model presented in this article, so there is a significant difference between the effect of the two models and the effect of this model. By analyzing the F1 values of the five models, the validity of the model is proved to be superior to the other four models.

Figure 6 represents the F1 value transformation trend of the five models in the text classification that iterate 20 times.

As can be seen from Figures 6 and Table 6, the F1 value of the model increases as the number of iterations value increases. The figure shows that the proposed model in this article is generally better than the other four models. The model in this article has only experienced small fluctuations with 11 and 14 iterations, with an increasing trend over the rest of the iterations. The GRU model and the LSTM model achieve good stability in the five models, increase with the number of iterations, and eventually stabilize. RNN model is less stable and floats more at 16 iterations. Overall, the recall rates of the five models increase with the number of iterations, and the F1 value of the model presented in this

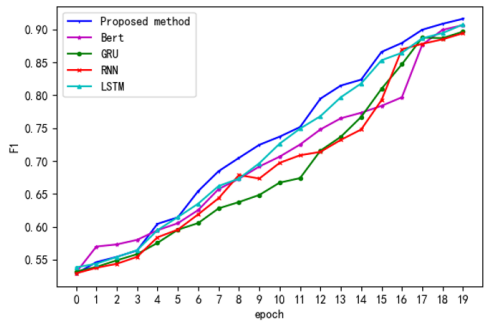

FIGURE 6. Model iterative graph (recall rate)

paper is better than that of the other models in each iteration, but the stability of the model needs to be further improved.

5. Epilogue

With the development of network technology in China, Weibo has become one of the most popular social tools for Internet users, through the emotional analysis of user comment text, it is possible to monitor the public opinion of the network in more real time, fast and accurate.

This article focuses on the short sentence structure and unobvious grammatical features of the Weibo comment text, proposed a two-input Adaboost model based on TextCNN and BiLSTM, The model can perform local feature extraction and global feature extraction on the text respectively, and then use the Adaboost algorithm for the fusion operation of the running results of the two models.

Experiments show that the accuracy, recall rate and F1 value of the model are obviously better than several traditional classification models, and the classification accuracy is improved.

References

[1] A. Huettner and P. Subasic, *Fuzzy Typing for Document Management*, ACL 2000 Companion Volume: Tutorial Abstracts and Demonstration Notes (2000), 26–27.

[2] D. Tang, F. Wei, B. Qin and T. Liu, *Building Large-Scale Twitter-Specific Sentiment Lexicon: A Representation Learning Approach*, Proceedings of coling 2014, the 25th International Conference on Computational Linguistics: Technical papers (2014), 172–182.

[3] C. Keyu and Z. He, *Study on Emotional Classification of Hotel Reviews Based on Emotion Dictionary*, Modern Computer (Professional Edition) (2017), no. 6, 3–6.

[4] E. Kouloumpis, T. Wilson and J. Moore, *Twitter Sentimnet Analysis: The Good the Bad and the OMG!*, Fifth International AAAI Conference on Weblogs and Cocial Media (2011), 538–541.

[5] P. D. Turney, *Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews*, Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics (2002), 417–424.
[6] B. Pang, L. Lee and S. Vaithyanathan, *Thumbs up? Sentiment classification using machine learning techniques*, Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing-Volume 10. Association for Computational Linguistics (2002), 79–86.
[7] Y. Kim, *Convolutional Neural Networks for Sentence Classification*, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014), 1746–1751.
[8] X. Wang, F. Wei, X. Liu, M. Zhou and M.Zhang, *Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach*, Proceedings of the 20th ACM International Conference on Information and knowledge Management (2011), 1031–1040.
[9] S. Pal, S. Ghosh and A. Nag, *Sentiment Analysis in the Light of LSTM Recurrent Neural Networks*, International Journal of Synthetic Emotions **9** (2018), no. 1, 33-39.
[10] J. Cross and L. Huang, *Incremental Parsing with Minimal Features Using Bi-Directional LSTM*, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (2016), 32–37.

Zhong-Shi Li
School of Economics and Management, Yanbian University, Yanji, Jilin, China

Lin He
College of Science major of applied statistics, Yanbian University, Yanji, Jilin, China

Wei-Jie Guo
College of Science major of applied statistics, Yanbian University, Yanji, Jilin, China

Zhe-Zhi Jin*
School of Economics and Management, Yanbian University, Yanji, Jilin, China. Corresbonding author