Combining Feature Selection Methods with BERT: An In-depth Experimental Study of Long Text Classification

Kai Wang, Jiahui Huang, Yuqi Liu, Bin Cao(✉), and Jing Fan

Zhejiang University of Technology, Hangzhou, China
{wangkai,huangjh,liuyuqi,bincao,fanjing}@zjut.edu.cn

Abstract. With the introduction of BERT by Google, a large number of pre-training models have been proposed. Using pre-training models to solve text classification problems has become the mainstream. However, the complexity of BERT grows quadratically with the text length, hence BERT is not suitable for processing long text. Then the researchers proposed a new pre-training model XLNet to solve the long text classification problem. But XLNet requires more GPUs and longer fine-tuning time than BERT. To the best of our knowledge, no attempt has been done before combining traditional feature selection methods with BERT for long text classification. In this paper, we use the classic feature selection methods to shorten the long text and then use the shortened text as the input of BERT. Finally, we conduct extensive experiments on the public data set and the real-world data set from China Telecom. The experimental results prove that our methods are effective for helping BERT to process long text.

Keywords: Text classification · Long text · BERT · Feature selection

1 Introduction

Text classification has been widely studied and used in many real applications, such as e-mail filtering [7], news classification [1], complaint classification [26], etc. Traditional machine learning methods have been widely used in text classification, such as Logistics Regression (LR) [8,9], Support Vector Machine (SVM) [10,17] and Random Forest (RF) [3]. However, these methods need to construct feature engineering in advance and use One-Hot encoding to represent words. The disadvantage of this is that the relationship between words and the semantic connection between contexts cannot be obtained. With the development of deep learning and the powerful representation capabilities of word vectors, the use of neural networks to solve text classification problems has become the mainstream. The neural networks do not need to construct feature engineering and can capture the semantic information of the text, have achieved excellent classification results. Researchers have proposed many neural network models based on

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 349, pp. 567–582, 2021.
https://doi.org/10.1007/978-3-030-67537-0_34

CNN and RNN with good classification results, such as TextCNN [11], TextRNN [15,30], Bi-LSTM [31] and TextRCNN [14].

Subsequently, the pre-training model BERT [6] was proposed, and it has achieved great results in many NLP tasks. The network architecture of BERT is composed of multiple layers of Transformers [25], which are based on Multi-head self-attention [25]. Since the computational complexity and space complexity of Transformers are $O(n^2)$, where n is the length of the input sequence, Transformers require huge time and GPU memory to process long sequences, thus BERT requires huge GPU memory for long text. The input text length of BERT is limited to a fixed length L, generally $L = 128,\ 256,\ or\ 512$. However, in long text classification tasks, the length of many texts will exceed L. Take the real-world data of China Telecom as an example, as shown in Fig. 1. As can be seen from Fig. 1, when $L = 512$, only a small number of texts are longer than L. When $L = 256$, there are more than 10,000 texts with the text length exceeding L. When $L = 128$, more than half of the texts are longer than L. Limited by the GPU, L is usually 128 or 256 in practice, which means that BERT cannot obtain all the information of many texts.

Researchers have proposed many solutions to this shortcoming that Transformer cannot handle long sequences. Child et al. [4] proposed the Sparse Transformer, which reduces the complexity of Multi-head self-attention through factorization. Dai et al. [5] proposed Transformer-XL to cut long text into fragments and establish connections between fragments to solve the problem of long dependence. Rae et al. [21] proposed a Compression Transformer based on Transformer-XL. Then another pre-training model XLNet [29] which is based on Transformer-XL has been proposed and has become one of the most effective models. XLNet outperforms BERT in many NLP tasks and solves the problem that BERT cannot handle long texts. However, XLNet requires more GPUs and longer fine-tuning time than BERT [16].

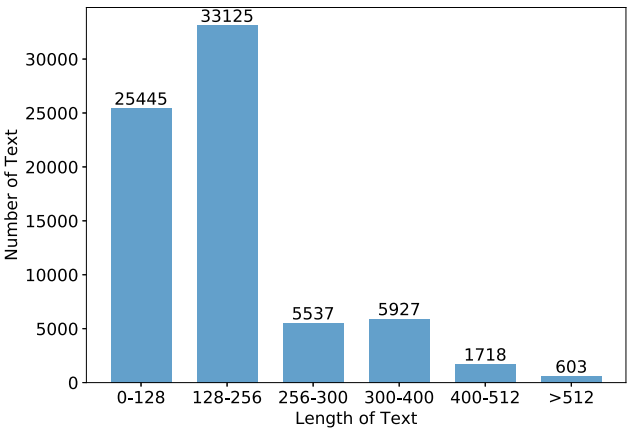


Fig. 1. Text length distribution of China Telecom text data (one month)

In this paper, we adopt six feature selection methods to improve the performance of BERT on long texts. The first four methods are traditional feature selection methods, including TF-IDF [22], TextRank [18], Mutual Information (MI) [13] and χ^2-Statistic (CHI) [23]. We use those four feature selection methods to shorten the length of the text while retaining the key information of the text, and then put the shortened text into the BERT. Considering that the key points of the text may appear at the end of the text, we put forward the next two simple truncation methods: Tail-only and Head+Tail. Tail-only means only to keep the last L tokens of the text, and Head+Tail means to keep the first $L/2$ tokens and the last $L/2$ tokens. Besides, we add XLNet as our baseline. Finally, we conduct extensive experiment on the public data set and real-world data set from China Telecom, experimental results show that our methods are effective.

The contributions of this paper can be concluded as follows:

- We put forward a very common problem in reality that BERT is not suitable for processing long text.
- We propose the method that combines feature selection methods and BERT to process long text. To the best of our knowledge, no attempt has been done before.
- We conduct extensive experiments based on public data set and real-world data set, experimental results prove the effectiveness of our methods.

The rest of this paper is organized as follows. In Sect. 2, we present the related work. Section 3 outlines the preliminaries. In Sect. 4, we describe our experimental results on real-world data set. Finally, we conclude our paper in Sect. 5.

2 Related Works

There are basically two ways to solve the problem that BERT cannot handle long text. The first is to split the input text into several short sentences, and use BERT to process each short sentence separately. The second method is to improve the Transformer to reduce the complexity of the Transformer so that BERT can handle long text.

Yang et al. [28] segmented the long text into several sentences, then put each sentence into the BERT, score each sentence, then score the long text according to the sentence score, and finally process the long text. Pappagari et al. [20] segmented the input text into several smaller chunks, and then put them into the model like BERT to obtain the representation of each chunk, and finally use either a recurrent LSTM network or another Transformer to perform the actual classification. However, this method cannot obtain the semantic information between the chunks.

Dai et al. [5] improved Transformer and proposed Transformer-XL. In order to get the ability to capture long-range dependencies, Transformer-XL proposed a segment-level recurrence mechanism and introduced a memory module, which

can keep the connection between the segments. They also proposed a novel relative positional embedding scheme which outperforms the Transformer's original absolute positional system. Sukhbaatar et al. [24] improve the calculation efficiency of the Transformer, they proposed an adaptive attention span that allows the model to adaptively select the context length for processing, thereby reducing the running time and memory usage of the Transformer. Rae et al. [21] proposed the Compressive Transformer, a simple extension to the Transformer which maps past hidden activations (memories) to a smaller set of compressed representations (compressed memories). Kitaev et al. [12] proposed Reformer, they replaced dot-product attention by one that used locality-sensitive hashing, changing the complexity from $O(L^2)$ to $O(LlogL)$. Besides, they adopted reversible residual layers. Beltagy et al. [2] proposed Longformer. The attention mechanism of Longformer is a drop-in replacement for the standard self-attention and combines a local windowed attention with a task motivated global attention.

3 Preliminaries

In this section, we will briefly introduce our feature selection methods: TF-IDF, TextRank, Mutual Information (MI) and χ^2-Statistic (CHI), finally, we will introduce the BERT.

3.1 TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is used to measure the importance of words to sentences. TF refers to the number of times a given word appears in the texts, IDF means that the fewer texts that contain the word, the greater the IDF, indicating that the word has a good ability to distinguish categories. TF and IDF can be calculated with the following formula:

$$TF_w = \frac{n_w}{N},\ IDF = log(\frac{D}{d_w + 1}) \tag{1}$$

where n_w means the number of the word w, N means the number of all the words, D means the number of all the texts, d_w means the number of texts which contain the word w. Finally, TF-IDF can be calculated with the following formula:

$$TF - IDF = TF * IDF \tag{2}$$

3.2 TextRank

TextRank builds a network through the adjacent relationship between words, and then uses PageRank [19] to iteratively calculate the rank value of each node, and sort the rank value to get the keyword. The PageRank formula is as follows:

$$PR(V_i) = (1 - d) + d * \sum_{j \in In(V_i)} \frac{1}{Out(V_j)} PR(V_j) \tag{3}$$

where $PR(V_i)$ represents the rank value of node V_i, $In(V_i)$ represents the set of predecessor nodes of node V_i, $Out(V_j)$ represents the set of successor nodes of node V_j, d is the damping factor for smoothing.

TextRank keyword extraction for sentences requires only word segmentation of the text, each word is a node. At the same time, assuming that each word is only related to n words in its vicinity, and an undirected edge corresponding to n words in its vicinity is connected, so that a graph is formed and the Rank value of each word can be calculated.

3.3 χ^2-Statistic

χ^2-Statistic (CHI) is a measure of the degree of association between the feature item t_i and category C_j, and it is assumed that the relationship between t_i and C_j conforms to the χ^2 distribution with first-order degrees of freedom. The higher the χ^2 statistic of the feature for a certain class, the greater the correlation, the more category information it carries.

Table 1. The relation between feature item and category

Feature item	Category	
	C_j	$\sim C_j$
t_i	A	B
$\sim t_i$	C	D

As shown in Table 1, Let N be the total number of texts, A represents the number of texts belonging to category C_j and containing t_i, B represents the number of texts not belonging to category C_j but containing t_i, C represents the number of texts belonging to C_j but not containing t_i, and D represents the number of texts that neither belong to C_j category nor contain t_i. The CHI value of t_i and C_j can be calculated according to the formula:

$$\chi^2(t_i, C_j) = \frac{N \times (A \times D - C \times B)^2}{(A + C) \times (B + D) \times (A + B) \times (C + D)} \tag{4}$$

For multi-class problems, suppose M is the number of all categories, we calculate the CHI value of t_i for each category separately, and then take the maximum value:

$$\chi^2_{MAX}(t_i) = max_{j=1}^{M} \chi^2(t_i, C_j) \tag{5}$$

3.4 Mutual Information

Mutual Information (MI) is similar to χ^2-Statistic, the greater the MI, the greater the degree of co-occurrence of feature t_i and category C_j. Similar to

the definition of χ^2-Statistic, the mutual information between t_i and C_j can be calculated by the following formula:

$$I(t_i, C_j) = log(\frac{A \times N}{(A + C) \times (A + B)}) \tag{6}$$

Similar to the CHI value, Multi-class mutual information can be obtained by the following formula:

$$I_{MAX}(t_i) = max_{j=1}^{M}[P(C_j) \times I(t_i, C_j)] \tag{7}$$

3.5 BERT

BERT (Bidirectional Encoder Representations from Transformers), a pre-trained model whose goal is to use large-scale unlabeled training corpora to obtain a textual representation containing rich semantic information, and achieved good results in many NLP tasks. The main structure of BERT is Transformer. Transformer is formed by stacking several encoders and decoders, abandoning traditional CNN and RNN, and the entire network structure is composed of the attention mechanism. The input of BERT combines three feature embeddings: token embedding, position embedding, and segment embedding. Token embedding converts each word into a fixed-dimensional vector. Segment embeddings are used to distinguish two sentences. Position embedding encodes the position information of the word into a feature vector. BERT is a multi-task model, and its task is composed of two self-supervised tasks, namely MLM and NSP. MLM refers to masking some words randomly from the input expectation during training, and then predicting the word through the context. The task of NSP is to determine whether sentence B is the bottom of sentence A.

Although BERT has achieved good results in many NLP tasks, due to the defects of the Transformer, BERT has become inadequate when facing long texts.

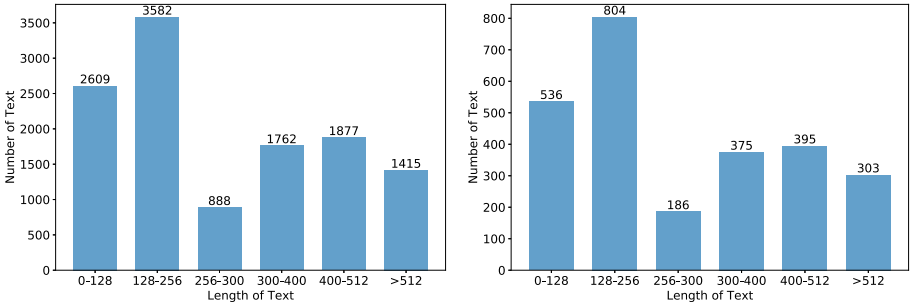
4 Experimental Evaluation

In this section, we evaluate the performance of each method based on different fixed length L. We first shorten the text whose text length larger than L by different feature selection methods, and then use the shortened text as the input of BERT. Our feature selection methods including TF-IDF, TextRank, MI, CHI, Tail-only, Head+Tail. In addition, we add XLNet as our baseline. For the experimental results, our evaluation metrics including Accuracy, Precision, Recall, F1-score, Hamming-loss and ROC curve & AUC. In order to prove the effectiveness of our methods, we conduct experiments on both public data set and real-world data set. All experiments in our work are evaluated on a computer with Intel Core(TM) i9-9940X 3.30 GHz CPU and RTX 2080Ti graphics and Python 3.6.10. We conduct the experiment of XLNet on 3 GPUs and other experiments on 1 GPU.

4.1 Datasets

We employ two groups of data sets namely IFLYTEK[27] and DianXin.

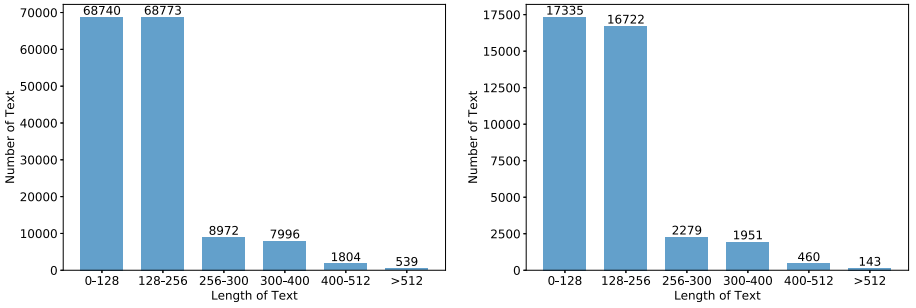
IFLYTEK is a collection of application description text from the famous Chinese AI company iFLYTEK. IFLYTEK has a total of 190 categories, the training set has 12,133 texts, and the test set has 2599 texts. The text length distribution of IFLYTEK is shown in Fig. 2. Considering that 512 tokens can basically express the semantic information of the text, when L = 512, the experimental difference cannot be well reflected, so we conduct our experiments when $L = 256$ and $L = 128$ respectively.



(a) Training set (b) Test set

Fig. 2. Text length distribution of IFLYTEK

DianXin is a collection of real-world complaint texts from China Telecom. DianXin is divided into training set and test set, a total of 580 categories. The training set has 156,834 texts and the test set has 38,890 texts. The text length distribution of DianXin is shown in Fig. 3. From the figure we can see that there are nearly 12% of the texts whose length larger than 256, and there are more than 50% of the texts whose length larger than 128. Therefore, we conduct experiments when $L = 128$ and $L = 256$ respectively to prove that our methods are helpful for BERT to process long texts.



(a) Training set (b) Test set

Fig. 3. Text length distribution of DianXin

4.2 Evaluation Metrics

Confusion Matrix. Figure 4 is a confusion matrix of binary classification. TP (True Positive) indicates the number of samples that predict the positive class as a positive class. FN (False Negative) indicates the number of samples that predict the positive class as the negative class. FP (False Positive) indicates the number of samples that predict the negative class as the positive class. TN (True Negative) represents the number of samples that predict the negative class as a negative class.

In multi-classification, we use the *OVR* method, which means that when we perform classification tasks in a certain class, we treat all other classes as negative classes.

Confusion Matrix		True Label	
		Positive	Negative
Predictive Label	Positive	TP	FP
	Negative	FN	TN

Fig. 4. Confusion matrix

Accuracy. Accuracy is the overall evaluation of the model, and it is the most intuitive metric for evaluating the quality of the model. The accuracy is calculated in the binary classification as:

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}}$$

and in multi-classification is calculated as:

$$Accuracy = \frac{\sum \mathrm{TP}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}}$$

Precision. Precision represents the proportion of samples that are truly positive among all the samples predicted to be positive. The precision is calculated in the binary classification as:

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

and in multi-classification is calculated as:

$$Precision = \frac{1}{n} \sum_{i=1}^{n} P_i$$

Recall. Recall is the percentage of predicted positive classes among all positive classes. The recall is calculated in the binary classification as:

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

and in multi-classification is calculated as:

$$Recall = \frac{1}{n} \sum_{i=1}^{n} R_i$$

F1-Score. Precision and recall are a pair of contradictory quantities. When precision is high, recall tends to be relatively low, and when recall is high, precision tends to be relatively low. So in order to better evaluate the performance of the classifier, F1-Score is generally used as the evaluation standard to measure the comprehensive performance of the classifier. F1-Score is calculated as:

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Hamming-Loss. Hamming-loss is applicable to the problem of multi-classification. In short, it is to measure the loss between the predicted label and the real label, and the value is between 0 and 1. A loss of 0 indicates that the prediction result is exactly the same as the real result, and a loss of 1 indicates that the model is completely contrary to the result we want.

ROC Curve & AUC. ROC (Receiver Operating Characteristic) curve is a commonly used model evaluation method in binary classification. The abscissa of the ROC curve is the false positive rate (FPR), which is calculated as:

$$FPR = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

the ordinate of the ROC curve is the true positive rate (TPR), which is calculated as:

$$TPR = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

The greater the FPR, the more actual negative classes are predicted in the positive class, and the greater the TPR, the more actual positive classes in the positive class.

AUC (Area under Curve): The area under the Roc curve, between 0.1 and 1. As a numerical value, AUC can directly evaluate the quality of the classifier. The greater the value, the better the classifier.

4.3 Experimental Result of IFLYTEK

Due to the small scale of the IFLYTEK, we calculated the weighted-average precision, recall and F1-score. The experimental results when $L = 256$ are shown in Table 2. In terms of accuracy, recall and hamming-loss, the CHI+BERT method achieves the best results. On precision, CHI+BERT has almost the same result as BERT. On F1-sore, CHI+BERT also achieved results second only to XLNet. Overall, the CHI+BERT method outperforms BERT in four metrics, which means that the feature selection method is helpful for BERT to process long texts.

Table 2. Experimental results of IFLYTEK when $L = 256$

Experimental methods	Accuracy	Precision	Recall	F1-score	Hamming-loss
BERT	0.608	0.592	0.608	0.585	0.392
TF-IDF+BERT	0.595	0.565	0.594	0.570	0.405
TextRank+BERT	0.605	0.592	0.605	0.584	0.395
MI+BERT	0.600	0.573	0.600	0.576	0.400
CHI+BERT	**0.611**	0.589	**0.611**	**0.592**	**0.389**
Tail-only	0.586	0.567	0.586	0.562	0.414
Head+Tail	0.606	**0.593**	0.606	**0.586**	0.394
XLNet	0.598	**0.607**	0.598	**0.593**	0.402

Table 3. Experimental results of IFLYTEK when $L = 128$

Experimental methods	Accuracy	Precision	Recall	F1-score	Hamming-loss
BERT	0.601	0.587	0.601	0.582	0.399
TF-IDF+BERT	0.596	0.574	0.596	0.575	0.404
TextRank+BERT	0.594	0.578	0.594	0.574	0.406
MI+BERT	0.569	0.540	0.569	0.543	0.431
CHI+BERT	**0.606**	**0.587**	**0.606**	**0.585**	**0.394**
Tail-only	0.556	0.520	0.556	0.525	0.444
Head+Tail	**0.602**	0.581	**0.602**	0.580	**0.398**
XLNet	0.590	**0.598**	0.590	**0.588**	0.410

The experimental results when $L = 128$ are shown in Table 3. As can be seen from the table, the CHI+BERT method has achieved the best results in accuracy, recall and hamming-loss, and it is second only to XLNet and better than BERT in precision and F1-score. CHI+BERT method outperforms BERT in all metrics, which proves the effectiveness of feature selection.

In conclusion, On the IFLYTEK data set, the CHI+BERT method performs better than BERT, which to a certain extent shows that the combined feature selection methods help BERT to process long text.

4.4 Experimental Results of DianXin

In order to demonstrate the effectiveness of our method in reality, we implemented our experiments on the real-world data set DianXin. And we added the evaluation metric ROC & AUC. We calculate the macro-average precision, recall and F1-score.

The experimental results when $L = 256$ are shown in Table 4. It can be seen from Fig. 3 that when $L = 256$, about 88% of the texts are less than 256 in length, and only about 12% of the texts has feature selection operations. Therefore, the evolution metrics in Table 4 are very close. In terms of accuracy, the accuracy of CHI+BERT, Tail-only and XLNet surpasses BERT, and the gap between CHI+BERT and XLNet is very small. As for precision, CHI+BERT is the highest, next is TF-IDF+BERT. In terms of recall, XLNet has greatly improved, followed by TF-IDF+BERT. XLNet also achieved the best results in F1-score. As for hamming-loss, CHI+BERT and XLNet both have good performance.

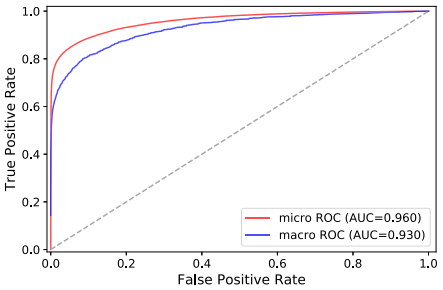
The ROC curve and AUC value of each method are shown in Fig. 5. For the two ROC curves, the results of each method are not much different, the macro curve of TF-IDF+BERT is better, and the micro curve of TextRank is better.

Overall, CHI+BERT and XLNet are the best two methods, surpassing BERT in 5 evaluation metrics, and the accuracy of the two is almost the same. The result proves that our methods are effective.

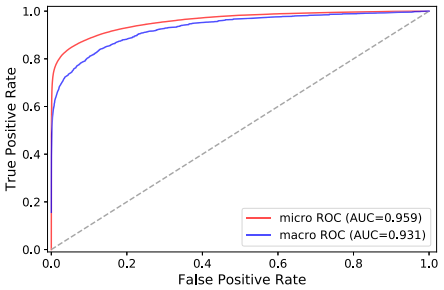
Table 4. Experimental results of DianXin when $L = 256$

Experimental methods	Accuracy	Precision	Recall	F1-score	Hamming-loss
BERT	0.593	0.408	0.354	0.368	0.407
TF-IDF+BERT	0.590	**0.414**	**0.356**	**0.369**	0.410
TextRank+BERT	0.591	**0.409**	0.353	0.365	0.409
MI+BERT	0.593	0.407	0.352	0.366	0.407
CHI+BERT	**0.597**	**0.415**	**0.355**	**0.369**	**0.403**
Tail-only	**0.594**	0.403	**0.355**	0.366	**0.406**
Head+Tail	0.591	0.401	0.352	0.363	0.409
XLNet	**0.599**	**0.409**	**0.380**	**0.383**	**0.401**

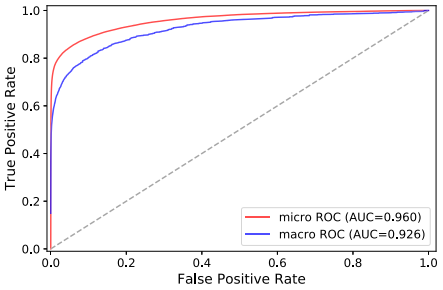
The experimental results when $L = 128$ are shown in Table 5. It can be seen from Fig. 3 that when $L = 256$, about 40% of the texts are less than 256 in length, and about 60% of the texts has feature selection operations. It is obvious from Table 5 that the results of most methods are better than BERT. The five evaluation metrics of TF-IDF+BERT, CHI+BERT, Tail-only, Head+Tail and



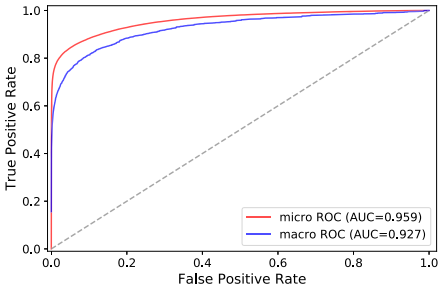
(a) BERT



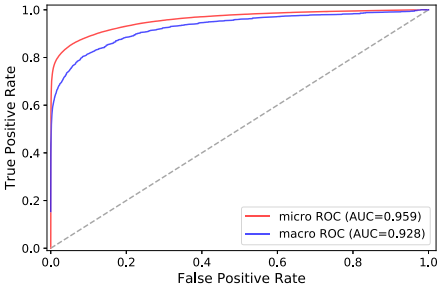
(b) TF-IDF+BERT



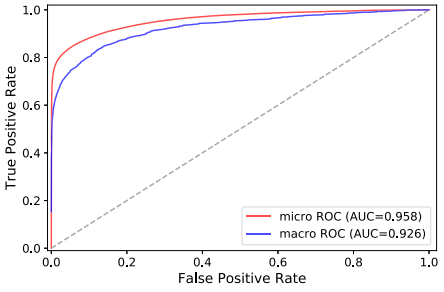
(c) TextRank+BERT



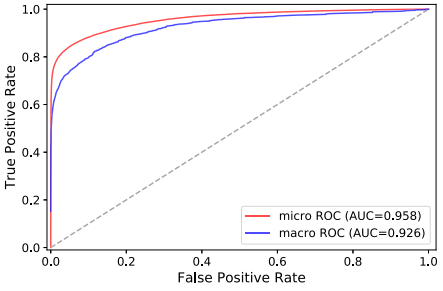
(d) MI+BERT



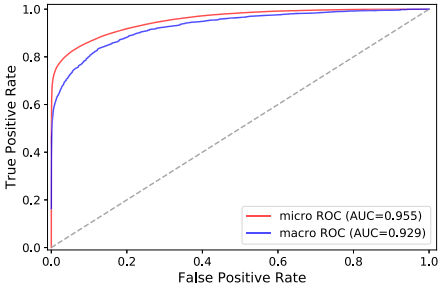
(e) CHI+BERT



(f) Tail-only

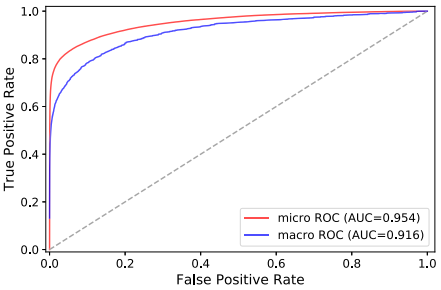


(g) Head+Tail

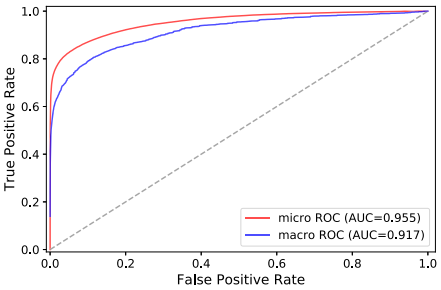


(h) XLNet

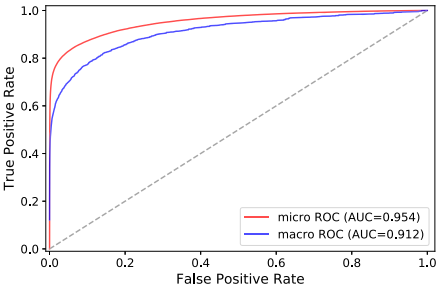
Fig. 5. Results of ROC & AUC ($L = 256$)



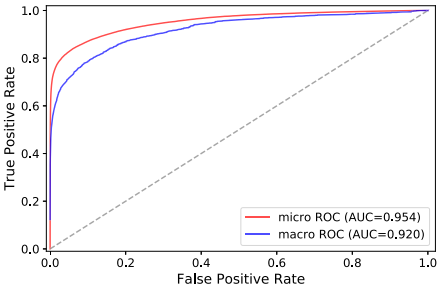
(a) BERT



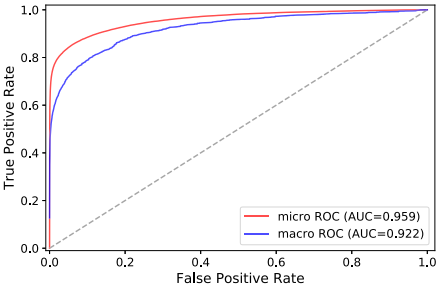
(b) TF-IDF+BERT



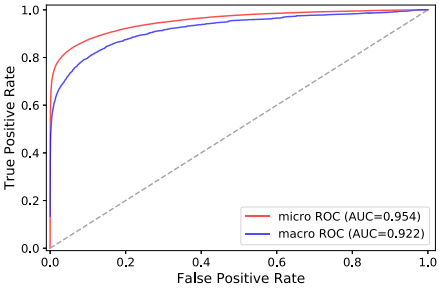
(c) TextRank+BERT



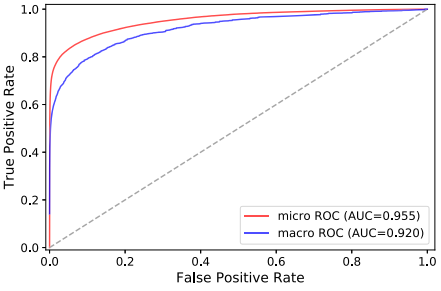
(d) MI+BERT



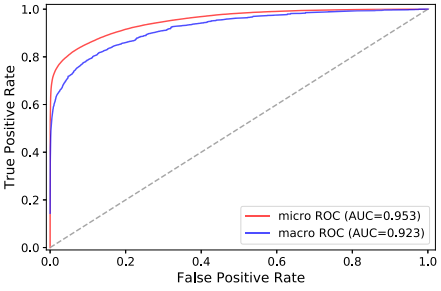
(e) CHI+BERT



(f) Tail-only



(g) Head+Tail



(h) XLNet

Fig. 6. Results of ROC & AUC ($L = 128$)

Table 5. Experimental results of DianXin when $L = 128$

Experimental methods	Accuracy	Precision	Recall	F1-score	Hamming-loss
BERT	0.557	0.363	0.315	0.325	0.443
TF-IDF+BERT	**0.564**	**0.385**	**0.334**	**0.344**	**0.436**
TextRank+BERT	0.555	**0.372**	0.311	**0.326**	0.445
MI+BERT	**0.559**	0.359	**0.316**	0.324	**0.441**
CHI+BERT	**0.574**	**0.373**	**0.319**	**0.332**	**0.426**
Tail-only	**0.562**	**0.374**	**0.323**	**0.334**	**0.438**
Head+Tail	**0.559**	**0.383**	**0.327**	**0.338**	**0.441**
XLNet	**0.579**	**0.390**	**0.354**	**0.361**	**0.421**

XLNet are all better than BERT. In terms of accuracy, the results of CHI+BERT and XLNet are very close, and both are 2% higher than BERT. On the three evaluation metrics of precision, recall and F1-score, the TF-IDF+BERT method achieved the best results except XLNet. As for Hamming-loss, both CHI+BERT and XLNet methods are much better than other methods.

When $L = 128$, the ROC curve of each method is shown in Fig. 6. Unlike $L = 256$, when $L = 128$, the ROC curves of other methods are better than those of BERT, and the ROC curve of CHI+BERT is the best.

In conclusion, combining the two experiments of $L = 128$ and $L = 256$, we can find that CHI+BERT always has a good performance except XLNet, better than BERT in almost all evaluation metrics, and is very close to XLNet in the most intuitive accuracy. This shows that our feature selection methods are very helpful for BERT processing long text.

5 Conclusion

In this paper, in order to improve the performance of BERT on long text classification. We propose a method that combines feature selection methods and BERT to process long text. We adopt six different feature selection methods: TF-IDF, TextRank, MI, CHI, Tail-only and Head+Tail. We conduct extensive experiments on public data set and real-world data set from China Telecom. Then we evaluate the experimental results by different evaluation metrics. Finally, the experimental results show that our methods could improve the long text classification performance of BERT and achieve the effect close to XLNet on China Telecom data set.

Acknowledgments. This research was sponsored by Zhejiang Lab (2020AA3AB05).

References

1. Ahmed, H., Traore, I., Saad, S.: Detecting opinion spams and fake news using text classification. Secur. Priv. **1**(1), e9 (2018)

2. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: the long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
3. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
4. Child, R., Gray, S., Radford, A., Sutskever, I.: Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509 (2019)
5. Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., Salakhutdinov, R.: Transformer-xl: attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
7. Diao, Y., Lu, H., Wu, D.: A comparative study of classification based personal e-mail filtering. In: Terano, T., Liu, H., Chen, A.L.P. (eds.) PAKDD 2000. LNCS (LNAI), vol. 1805, pp. 408–419. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45571-X_48
8. Genkin, A., Lewis, D.D., Madigan, D.: Large-scale Bayesian logistic regression for text categorization. Technometrics **49**(3), 291–304 (2007)
9. Hosmer Jr., D.W., Lemeshow, S., Sturdivant, R.X.: Applied Logistic Regression, vol. 398. Wiley, Hoboken (2013)
10. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0026683
11. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
12. Kitaev, N., Kaiser, Ł., Levskaya, A.: Reformer: the efficient transformer. arXiv preprint arXiv:2001.04451 (2020)
13. Kraskov, A., Stögbauer, H., Grassberger, P.: Estimating mutual information. Phys. Rev. E **69**(6), 066138 (2004)
14. Lai, S., Xu, L., Liu, K., Zhao, J.: Recurrent convolutional neural networks for text classification. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
15. Liu, P., Qiu, X., Huang, X.: Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101 (2016)
16. Liu, X., Wangperawong, A.: Transfer learning robustness in multi-class categorization by fine-tuning pre-trained contextualized language models. arXiv preprint arXiv:1909.03564 (2019)
17. Manevitz, L.M., Yousef, M.: One-class SVMs for document classification. J. Mach. Learn. Res. **2**(Dec), 139–154 (2001)
18. Mihalcea, R., Tarau, P.: TextRank: bringing order into text. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, pp. 404–411 (2004)
19. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the web. Tech. rep., Stanford InfoLab (1999)
20. Pappagari, R., Zelasko, P., Villalba, J., Carmiel, Y., Dehak, N.: Hierarchical transformers for long document classification. In: 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 838–844. IEEE (2019)
21. Rae, J.W., Potapenko, A., Jayakumar, S.M., Lillicrap, T.P.: Compressive transformers for long-range sequence modelling. arXiv preprint arXiv:1911.05507 (2019)
22. Ramos, J., et al.: Using TF-IDF to determine word relevance in document queries. In: Proceedings of the First Instructional Conference on Machine Learning, vol. 242, pp. 133–142, New Jersey (2003)

23. Satorra, A., Bentler, P.M.: A scaled difference chi-square test statistic for moment structure analysis. Psychometrika **66**(4), 507–514 (2001)
24. Sukhbaatar, S., Grave, E., Bojanowski, P., Joulin, A.: Adaptive attention span in transformers. arXiv preprint arXiv:1905.07799 (2019)
25. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
26. Wang, S., Wu, B., Wang, B., Tong, X.: Complaint classification using hybrid-attention GRU neural network. In: Yang, Q., Zhou, Z.-H., Gong, Z., Zhang, M.-L., Huang, S.-J. (eds.) PAKDD 2019. LNCS (LNAI), vol. 11439, pp. 251–262. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16148-4_20
27. Xu, L., et al.: Clue: a Chinese language understanding evaluation benchmark. arXiv preprint arXiv:2004.05986 (2020)
28. Yang, W., Zhang, H., Lin, J.: Simple applications of BERT for ad hoc document retrieval. arXiv preprint arXiv:1903.10972 (2019)
29. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V.: XLNet: generalized autoregressive pretraining for language understanding. In: Advances in Neural Information Processing Systems, pp. 5753–5763 (2019)
30. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489 (2016)
31. Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., Xu, B.: Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. arXiv preprint arXiv:1611.06639 (2016)