Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 7183207, 12 pages
https://doi.org/10.1155/2022/7183207

Research Article

A Multimodel-Based Deep Learning Framework for Short Text Multiclass Classification with the Imbalanced and Extremely Small Data Set

Jiajun Tong, Zhixiao Wang, and Xiaobin Rui

China University of Mining and Technology, School of Computer Science and Technology, Xuzhou, China

Correspondence should be addressed to Zhixiao Wang; zhxwang@cumt.edu.cn

Received 25 April 2022; Revised 26 August 2022; Accepted 23 September 2022; Published 6 October 2022

Academic Editor: Andrea Loddo

Text classification plays an important role in many practical applications. In the real world, there are extremely small datasets. Most existing methods adopt pretrained neural network models to handle this kind of dataset. However, these methods are either difficult to deploy on mobile devices because of their large output size or cannot fully extract the deep semantic information between phrases and clauses. This paper proposes a multimodel-based deep learning framework for short-text multiclass classification with an imbalanced and extremely small dataset. Our framework mainly includes five layers: the encoder layer, the word-level LSTM network layer, the sentence-level LSTM network layer, the max-pooling layer, and the SoftMax layer. The encoder layer uses DistilBERT to obtain context-sensitive dynamic word vectors that are difficult to represent in traditional feature engineering methods. Since the transformer part of this layer is distilled, our framework is compressed. Then, we use the next two layers to extract deep semantic information. The output of the encoder layer is sent to a bidirectional LSTM network, and the feature matrix is extracted hierarchically through the LSTM at the word and sentence level to obtain the fine-grained semantic representation. After that, the max-pooling layer converts the feature matrix into a lower-dimensional matrix, preserving only the obvious features. Finally, the feature matrix is taken as the input of a fully connected SoftMax layer, which contains a function that can convert the predicted linear vector into the output value as the probability of the text in each classification. Extensive experiments on two public benchmarks demonstrate the effectiveness of our proposed approach on an extremely small dataset. It retains the state-of-the-art baseline performance in terms of precision, recall, accuracy, and F1 score, and through the model size, training time, and convergence epoch, we can conclude that our method can be deployed faster and lighter on mobile devices.

1. Introduction

Text classification plays an important role in many practical applications [1]. Especially with recent breakthroughs in natural language processing (NLP) and text mining, it is widely used in many information processing systems, such as search engines [2, 3] and question answering systems [4, 5]. Generally, text classification can be divided into 3 steps: text preprocessing, feature extraction, and text representation, as shown in Figure 1.

In the real world, many datasets are unbalanced and extremely small, which are difficult to solve with traditional machine learning methods [6]. For example, questions about fires or accidents are usually less common than questions about consumer disputes about property services, but these data are too important to be ignored because of their scarcity and implications. Knowing how to efficiently and accurately classify those texts into specific categories according to the content of the corpus and improve the user experience has become an urgent problem to be solved.

To solve the abovementioned problem, most existing methods are based on the pretrained neural network, which can utilize a large amount of unlabeled data to learn common language representations and mine the relationship between context and target aspects [7]. Some of the most prominent methods are based on BERT [8]. Song et al. [9] explored the potential of utilizing BERT intermediate layers to enhance the performance of the fine-tuning part.

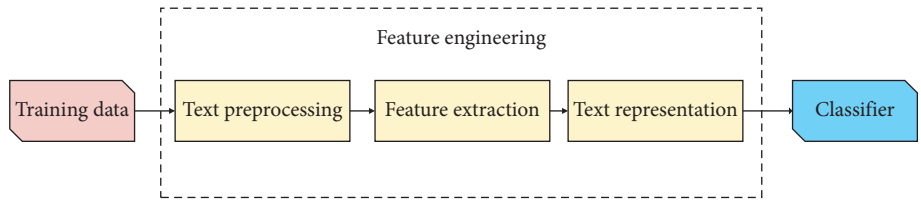

Figure 1: Overview of a text classification pipeline.

Wan and Li [10] proposed a combination model named BERT-CNN for the classification of candidate causal sentences. The number of encoder layers in BERT leads to not only a large number of training parameters but also a large output model size. Therefore, some distillation methods [11] have been proposed. Xiong and Yan [12] proposed a sentiment analysis model based on distillation bidirectional encoder representations from transformers combined with multiscale convolution. Adel et al. [13] presented an alternative event detection model based on DistilBERT and the Hunger Games search algorithm. DistilBERT is a small, fast, cheap, and light transformer model trained by distilling BERT base. It has 40% fewer parameters than BERT-base-uncased and runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark. The core idea of distillation is called teacher-student learning, where the student can reproduce the large model, and the teacher learns dark knowledge. Dark knowledge [14] includes two main parts: model compression and specialist networks. The model compression part transfer the language learned from the ensemble models into a single smaller model to reduce test computation, while the specialist networks training models specialized in a confusable subset of the classes to reduce the time to train an ensemble.

However, these methods are either difficult to deploy to mobile devices which usually generate a large amount of data. In the traditional computing architecture, data need to be transferred to the cloud for processing, which not only increases the bandwidth and data transmission time. Deploying pretrained models on mobile devices has natural advantages [15–17] over cloud computing. The entire workflow does not need to upload data to the cloud, can run offline, and fast response to improve data privacy. The pretrained model only needs a small amount of labeled data to complete the prediction and ensure the effectiveness of the prediction. Because of their output model size and require professional computing resources such as GPUs, or cannot fully extract the deep semantic information between phrases and clauses, leading to a low recall rate and accuracy.

To reduce the size of the model and maintain the classification accuracy of the model, this article proposes a multimodel-based deep learning framework for short text multiclass classification with an imbalanced and extremely small dataset. Our framework consists mainly of four layers. First, we use the pretrained DistilBERT as the encoder layer to obtain the context-sensitive dynamic word embeddings, including the token embeddings, position embeddings, and segment embeddings, as the input of the bidirectional LSTM network. Second, hidden features between phrases and clauses in the text are extracted through word-level and sentence-level LSTM networks to obtain the deep semantic information of sentences while stored as a feature matrix. To reduce the parameters and network complexity, we add a max-pooling layer to convert the feature matrix into a lower-dimensional matrix. Finally, for the multiclass classification task, the SoftMax layer normalizes multiple values obtained by the neural network to make the values between [0, 1] so that the results can be explained.

The main contributions of our paper can be summarized as follows:

(1) We built a multimodel-based framework combined with DistilBERT and BI-LSTM for the task of short text multiclass classification. Our method ensures accuracy and compresses the size of the output, which can be better applied to various mobile devices, such as smart appliances or smart cars.

(2) In the encoder layer, DistilBERT can convert the input to dynamic word embeddings, including token embeddings, segment embeddings, and position embeddings, which reduces the effective time and space complexities.

(3) The BI-LSTM is used as the feature extraction layer to extract the implicit features as fine-grained as possible through the word-level and sentence-level LSTM network to enhance the model's ability to solve polysemous word problems.

2. Related Work

As a fundamental topic in the NLP field, there is much research related to text classification. On the one hand, text categorization tasks are usually based on large amounts of annotated data, no matter whether they are supervised learning or self-supervised learning methods. Ghiassi et al. [18] present an integrated solution that combines a new clustering algorithm, with a domain transferrable feature engineering approach for Twitter sentiment analysis and spam filtering of YouTube comments. Kim et al. [19] propose a question-answer method to automatically provide users with infrastructure damage information from textual data. Stitini et al. [20] conclude that the linkage between contextual information and classification enhances and improves the recommendation results. Yang et al. [21] proposed a new automated defect text classification system (AutoDefect) based on a convolutional neural network (CNN) and natural language processing (NLP) using hierarchical two-stage encoders. Ma et al. [22] presented a level-by-level HMTC approach based on the bidirectional gated

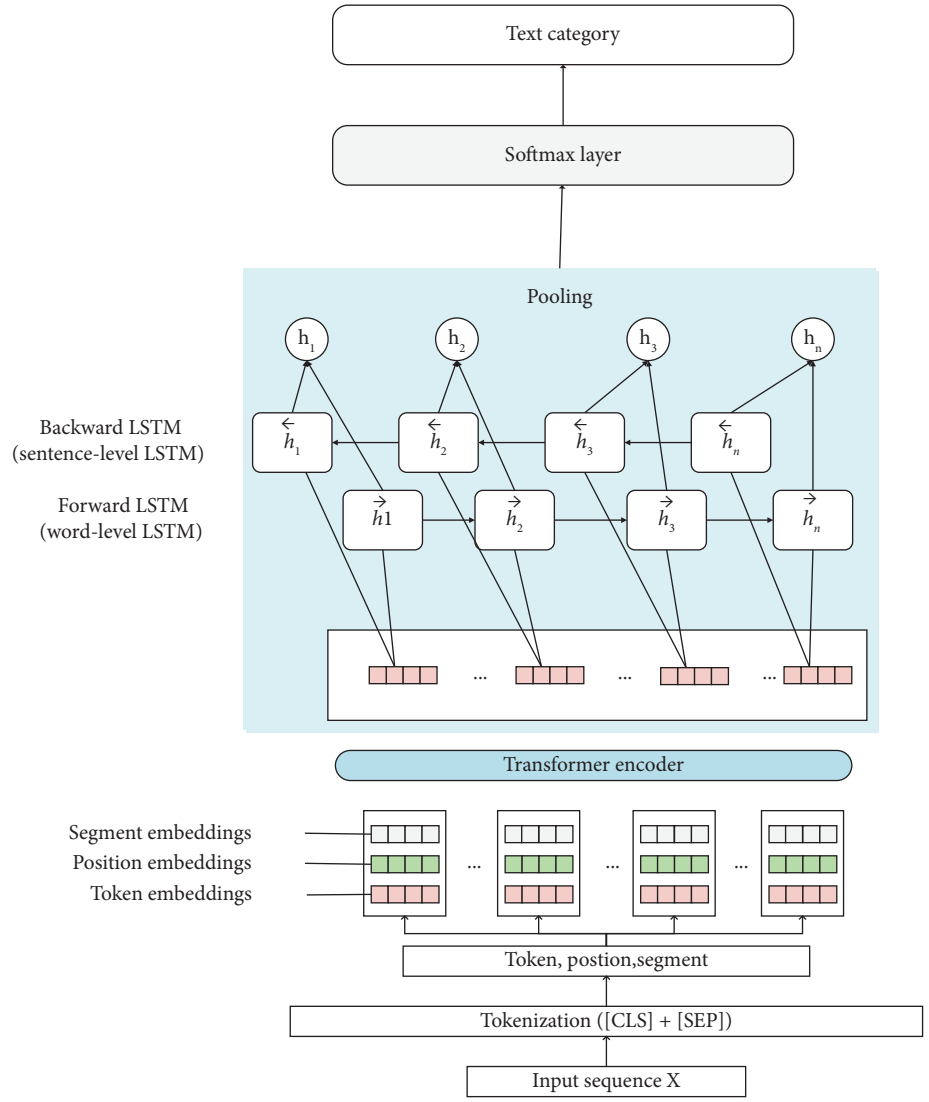

Figure 2: The framework based on multimodel-based deep learning.

recurrent unit network model together with hybrid embedding used to learn the representation of the text level-by-level. These methods explore the method of applying text classification to practical tasks.

However, many datasets are imbalanced and extremely small in the real world. Fakhar Bilal et al. [23] proposed a churn prediction model based on a combination of clustering and classification algorithms using an ensemble. Wang et al. [24] propose TkTC: a framework for top k text classification, where a novel loss function simultaneously considers the position of the ground truth label and the number of predictions. Zhu et al. [25] propose a simple short-text classification approach that makes use of prompt learning based on knowledgeable expansion, which can consider both the short text itself and the class name while expanding the label word space.

Although those short text classification methods based on the pretraining model can show good performance in the short text classification task, these models would generate a large output, which is unsuitable for mobile devices. Therefore, solutions based on DistilBERT are inspired. Andersen and Maalej [26] proposed a framework for the efficient, in-operation moderation of classifier output to maximize the accuracy and increase the overall acceptance of text classifiers. Chang et al. [27] developed a universal financial fraud awareness model to avoid these cases escalating to the level of crime. However, the above methods cannot guarantee the application performance on mobile devices, in which the memory space and computation resources are often limited.

In this article, we propose a multimodel-based framework combining the advantages of DistilBERT and BI-LSTM to reduce the model size in small dataset tasks while maintaining the performance of classification.

3. DistilBERT BI-LSTM Predictor

In this section, we propose a short-text classification framework based on multi-model learning combined with distillation encoder representations with bidirectional

LSTM, which trains tasks in a deep learning model, and the deep learning model combines neural networks with different structures to benefit from it. First, this model inputs texts into the encoder layer to obtain segment embeddings, position embeddings, and token embeddings. Second, the batched embeddings are sent into the forward LSTM to extract the word-level features, and the backward LSTM to extract the sentence-level features. Since the BI-LSTM extracts high-dimensional features, we adopt the max-pooling operation to sharpen the feature matrix and finally output the classification results through the Soft-Max layer. The overall structure of the model is shown in Figure 2, and the rest of this section will introduce this method in detail.

3.1. Encoder Layer. We remove the first input layer and have 6 encoder layers of DistilBERT [11]. DistilBERT was proposed by Sanh et al. [11] which is a small, fast, cheap, and light transformer model trained by the distilling BERT base. It has 40% fewer parameters than BERT-base-uncased and runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark. The main idea of distillation is to approximate the full output distributions of the BERT model using a smaller model such as DistilBERT. Thus, the number of transformer layers (encoders) in the BERT base (12 layers) has been reduced to six. The first token (CLS) vector of each encoding layer can be used as a sentence vector. We can abstractly understand that the shallower the encode layer, the better the sentence vector. It represents low-level semantic information, and the deeper it is, it represents high-level semantic information. We aim to obtain semantic features while preserving relevant word features. The specific method of the model is to use the CLS vectors of layers 1 to 6 as input of the LSTM for classification.

Semantic representation is assumed to be the contextual embedding learned by the token [CLS], which serves as the semantic representation of the input tweet X. The task is formulated as a multiclass classification problem. Therefore, the probability of X being classified as class c (i.e., an event) is predicted as the SoftMax function used in.

$$\Pr(c|X) = \text{Softamax}(\text{WT}\bullet X), \tag{1}$$

where W is the weight matrix learned during the fine-tuning of the pretrained model used during the initialization of the feature extractor model. It is worth noting that the first five transformer layers in the pretrained model are not trainable. We only fine-tuned the last transformer layer (encoder) of the pretrained model and replaced the classification layer with two fully connected layers for feature extraction and classification.

3.2. Word Level LSTM Network Layer. In the short text classification task, each input unit may have different degrees of impact on the final classification results. The LSTM network can filter the information from the input units. Based on the RNN, the network adds gates for selecting information, including an input gate, output gate, forgetting gate, and a memory unit to store information and update information through different gates.

To capture as much fine-grained classification information as possible in a small amount of text, this paper extracts and represents the semantic features of sentences through word- and sentence-level LSTM networks. The updating process of the word-level LSTM network is as follows: in time step t, the forgetting gate f_t^w will first judge which information can be forgotten according to the input x_t of the current time and the output h_{t-1}^w of the previous time, for example, the root units "##ing," "##ed" and crown words segmented. This can be described as follows:

$$f_t^w = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1}^w + b_f\right). \tag{2}$$

where W_* represents the weight matrix and b_* presents the bias term, all of which are network parameters to be learned; σ activates the sigmoid function.

Then, the input gate i_t^w will determine which information has a great impact on the classification label and needs to be updated. For example, some words have a great impact on the classification, such as "result," "case" and "great." The network will create a new memory cell candidate value $\widetilde{C}_t^w$ through the function. It can be described as follows:

$$\begin{aligned} i_t^w &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1}^w + b_i\right), \\ \widetilde{C}_t^w &= \tan h\left(W_{xc}x_t + W_{hc}h_{t-1}^w + b_c\right). \end{aligned} \tag{3}$$

When updating the memory cell C_t^w, it is necessary to combine the forget gate f_t^w and the memory cell C_{t-1}^w at the last time for information screening. The information with the quantity product $f_t^w \bullet C_{t-1}^w$ close to 0 will be discarded and added with $i_t^w \bullet \widetilde{C}_t^w$ to obtain the updated memory cell C_t^w to store the latest information. C_t^w can be described as follows:

$$C_t^w = f_t^w \bullet C_{t-1}^w + i_t^w \bullet \widetilde{C}_t^w. \tag{4}$$

Finally, the output gate o_i^w combines the memory unit C_t^w and the last time output h_{t-1}^w with the current time input x_t to calculate the word-level time step t as the output h_t^w. The calculation method is as follows:

$$\begin{aligned} o_i^w &= \sigma\left(W_{xo}x_t \bullet W_{ho}h_{t-1}^w + b_o\right), \\ i_t^w &= o_i^w \times \tanh\left(C_t^w\right). \end{aligned} \tag{5}$$

3.3. Sentence Level LSTM Network Layer. For the feature information output from the pretrained model, we use another LSTM to extract its fine-grained semantics and representation process. For the clause $S = \{s_1, \ldots, s_q, \ldots, s_n\}$ in sentence $s_q = \{w_{q1}, \ldots, w_{qp}, \ldots, w_{qm}\}$, the implicit feature h_{sq}^w obtained by the word-level LSTM network and the lexical feature B_{sq} obtained by BERT are spliced as the semantic feature representation h_{sq}^w of the clause, that is,

$$\begin{aligned} h_{sq}^s &= \left[(1-\lambda)B_{sq}, \lambda h_{sq}^w\right] q \in [1, n], \\ h_{sq}^w &= \text{LSTM}\left(w_{qp}\right) q \in [1, m], \end{aligned} \tag{6}$$

where λ represents the weight. Taking sentence $S = \{s_1, \ldots, s_q, \ldots, s_n\}$ as a sequence composed of n clauses, its feature vector is expressed as follows:

$$s = \begin{cases} h_{s1}^{w}, & n = 2, \\ [h_{s1}^{s}, h_{s2}^{s}, \ldots, h_{sn}^{s}] & n \geq 2. \end{cases} \tag{7}$$

The feature vector is updated through the sentence-level LSTM network to obtain the long sentence semantic representation h_t, and its process is similar to the word-level LSTM network, that is,

$$h_t = \mathrm{LSTM}(S(h_{st}^{s}))t \in [1, n]. \tag{8}$$

3.4. Max Pooling Layer. To improve the sharpening effect of pooling, we adopt the max-pooling operation. Because the BI-LSTM will eventually be connected to the fully connected layer and the number of neurons needs to be determined in advance, if the input length is uncertain, it is difficult to design its network structure. We use max-pooling to process the input X of uncertain length into a fixed-length input. Each filter takes only one value through the pooling operation. The number of neurons in the pooling layer corresponds to the number of filters so that the number of neurons in the eigenvector can be fixed. After the pooling operation, 2D or 1D arrays are usually converted to a single value, which can also reduce the number of parameters of a single filter or the number of hidden layer neurons for later use in the convolution layer or fully connected hidden layer.

3.5. SoftMax Layer. The output of the sentence-level LSTM network is transmitted to the full connection layer as the last semantic feature of the sentence, and the sentence classification result is obtained through the normalization operation of the SoftMax function. $p(l_i|h)$ presents the probability of text S in the ith classification. Its calculation method is as follows:

$$p(l_i|h) = \mathrm{Softmax}(h_i)_i = \frac{\exp(w_i h_t + b_i)}{\sum_{j=1}^{m} \exp(w_j h_t + b_j)}, \tag{9}$$

where w_* and b_* represent the weight matrix and offset term, respectively, and m represents the total number of tags. In this paper, the gradient descent algorithm is used to optimize the model and the cross entropy function is used to calculate the model loss and update the model parameters. The calculation method of loss function L is as follows:

$$L = -\frac{1}{m} \sum_{i=1}^{m} y_i \log(p(l_i|h)) + \varphi \|w_*\|^2, \tag{10}$$

where y_i represents the value of the digital vector of the real label of the text in the i dimension and φ represents the L2 regularization parameter.

4. Experiments

4.1. Dataset and Metrics. We use three different datasets to evaluate our methods. The HUFF datasets contain approximately 200k news headlines from 2012 to 2018 obtained from HuffPost. This dataset could be used to identify tags for untracked news articles or to identify the type of language used in different news articles. The COVID-Q [28] dataset has 1,690 questions about COVID, which are labeled into 16 unique categories. Considering that many text classification datasets need at least tens of thousands, this is a very small dataset. Finally, we evaluated it on a private dataset to see how it performed on a real-world application.

The notions of precision, recall, and F measures can be applied to each label independently in the multiclass task. Especially for an imbalanced classification problem with more than two classes, precision is calculated as the sum of true positives across all classes divided by the sum of true positives and false positives across all classes, where y represents the set of predicted(sample, label) pairs; $\hat{y}$ represents the set of true(sample, label) pairs; L is the set of label pairs; S represents the set of label pairs; y_s represents the subset of y with sample s, $y_s := \{(s', l) \in y | s' = s\}$; y_l represents the subset of y with label l. Similarly, $\hat{y}_s$ and $\hat{y}_l$ are subsets of $\hat{y}$. The F measurements can be defined as follows:

$$F_\beta(A, B) := \left(1 + \beta^2\right) \frac{P(A, B) \times R(A, B)}{\beta^2 P(A, B) + R(A, B)}, \tag{11}$$

where $P(A, B) := |A \cap B|/|A|$ for some sets $A$and B; $R(A, B) := |A \cap B|/|A|$ (Conventions vary on handling $B = \varnothing$; this implementation uses $R(A, B)$: $= 0$, and similar for P).

To give different weights to each type, we use weighted precision, recall, and f1-score. **Precision** is calculated as the sum of true positives across all classes divided by the sum of true positives and false positives across all classes. Then, we define precision as follows:

$$\frac{1}{\sum_{l \in L} |\hat{y}l|} \sum_{l \in L} |\hat{y}l| P(yl, \hat{y}l). \tag{12}$$

For example, we may have an imbalanced multiclass classification problem where the majority class is the negative class, but there are two positive minority classes: class 1 and class 2. Precision can quantify the ratio of correct predictions in both positive classes. **Recall** is calculated as the sum of true positives in all classes divided by the sum of true positives and false negatives across all classes. The recall is calculated as follows:

$$\frac{1}{\sum_{l \in L} |\hat{y}l|} \sum_{l \in L} |\hat{y}l| R(yl, \hat{y}l). \tag{13}$$

We did not calculate an overall F1 score. Instead, we calculate the **F1 score** per class in a one-vs-rest manner. In this approach, we rate the success of each class separately, as if there are distinct classifiers for each class. The F1 score is defined as follows:

$$\frac{1}{\sum_{l \in L} |\hat{y}l|} \sum_{l \in L} |\hat{y}l| F_\beta(yl, \hat{y}l). \tag{14}$$

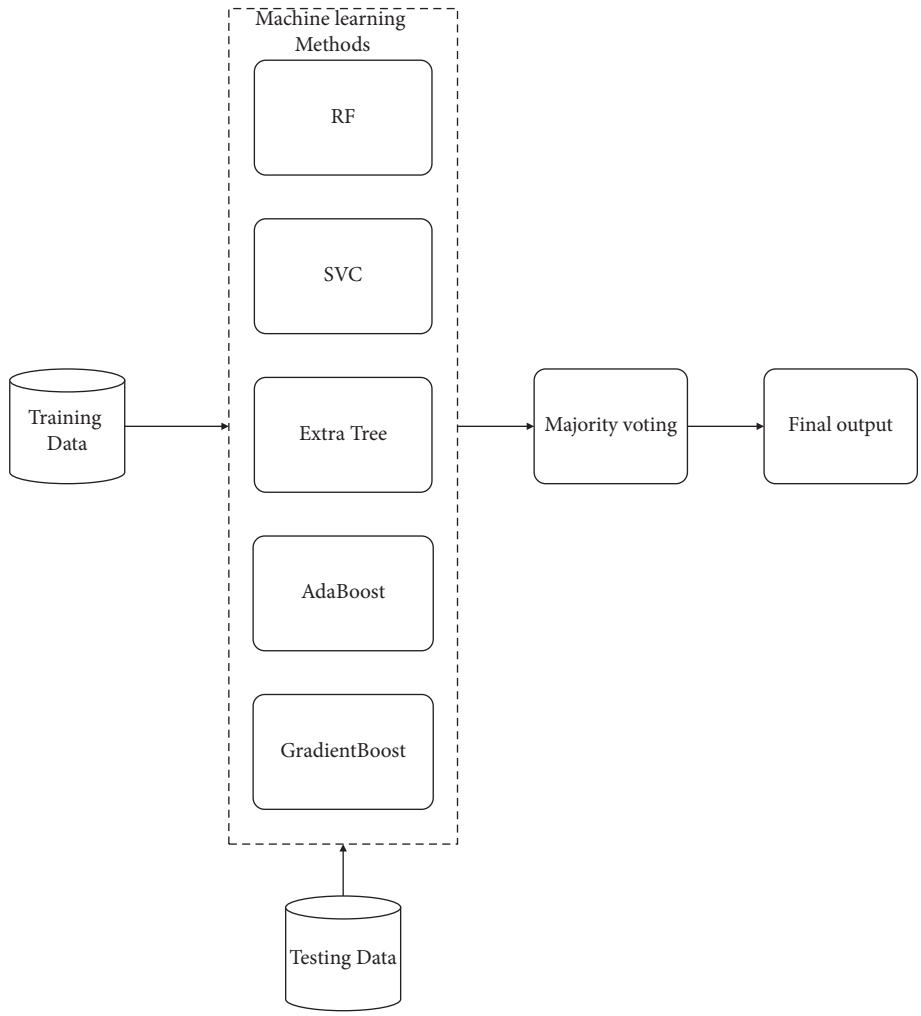

Figure 3: Five classifiers combined with the Ensemble model.

4.2. Results and Discussion. According to the method in Section 3, we construct a model and train the model on three different datasets. We verified the ensemble learning [29] and multimodel methods [30] on two public datasets. Moreover, to verify the performance of our model on small-scale datasets without changing the distribution of data, since there are approximately 200k lines of data in HUFF, we randomly extract 5% of the data from each type of data. At the same time, we deliberately keep the long tail problem for the data; that is, we do not limit the number of samples from each classification but extract them naturally from the dataset. We segment the dataset according to 64% of the training set, 16% of the verification set, and 20% of the test set.

We tried two different ways to mitigate the effects of imbalance for each approach. For the ensemble model part, we integrate algorithmic tree models like the random forest that are relatively insensitive to data skew. For the DistilBERT + BI-LSTM model part, we set the focal loss function, which focuses on adding weight to the corresponding loss of samples according to the difficulty of sample discrimination. That is, we add a smaller weight a_1 to easily distinguishable samples, and add a larger weight a_2 to hard distinguishable samples.

4.2.1. Comparison with Ensemble Modeling. Ensemble learning achieves better prediction performance by training multiple classifiers and combining these classifiers. The result of ensemble learning is usually better than a single model, which can reduce overfitting while improving accuracy, and the greater the difference between models (diversity), the more significant the improvement effect. This difference can be reflected in data, features, models, parameters, etc.

Encouraged by the good performance of the SVM model in the short-text classification task, we built an ensemble model to enhance this method, which compared 10 popular classifiers to evaluate the mean accuracy of each of them by a stratified k-fold cross-validation procedure. Thus, for the fine-tuning part, we performed a grid search optimization for 5 classifiers. Finally, we choose a voting classifier to combine the predictions, as shown in Figure 3.

Considering that the correlation between base models should be as small as possible, the performance gap should not be too large. We chose the SVC, AdaBoost, RandomForest, ExtraTrees, and GradientBoosting

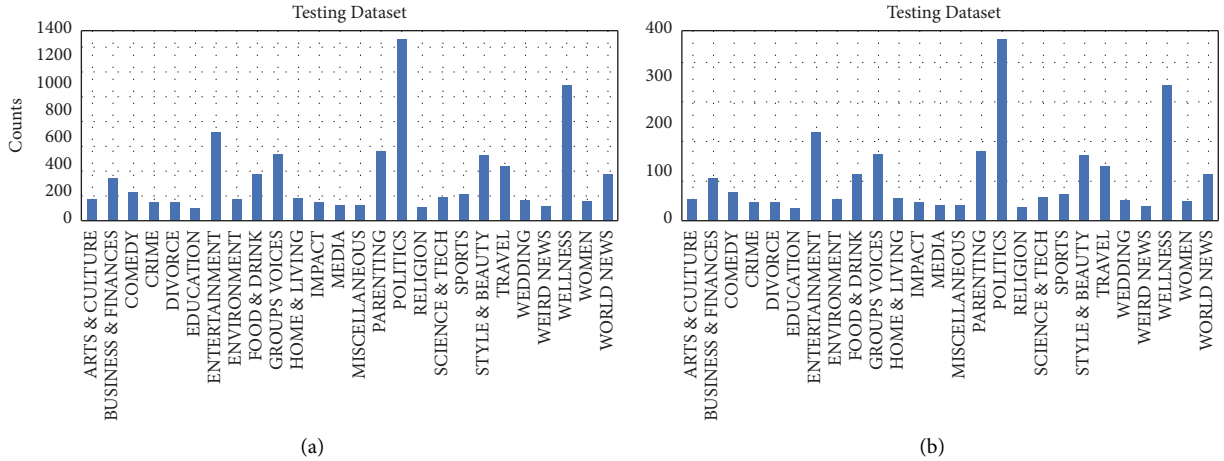

Figure 4: The distribution of HuffPost data. (a) Training dataset. (b) Testing dataset.

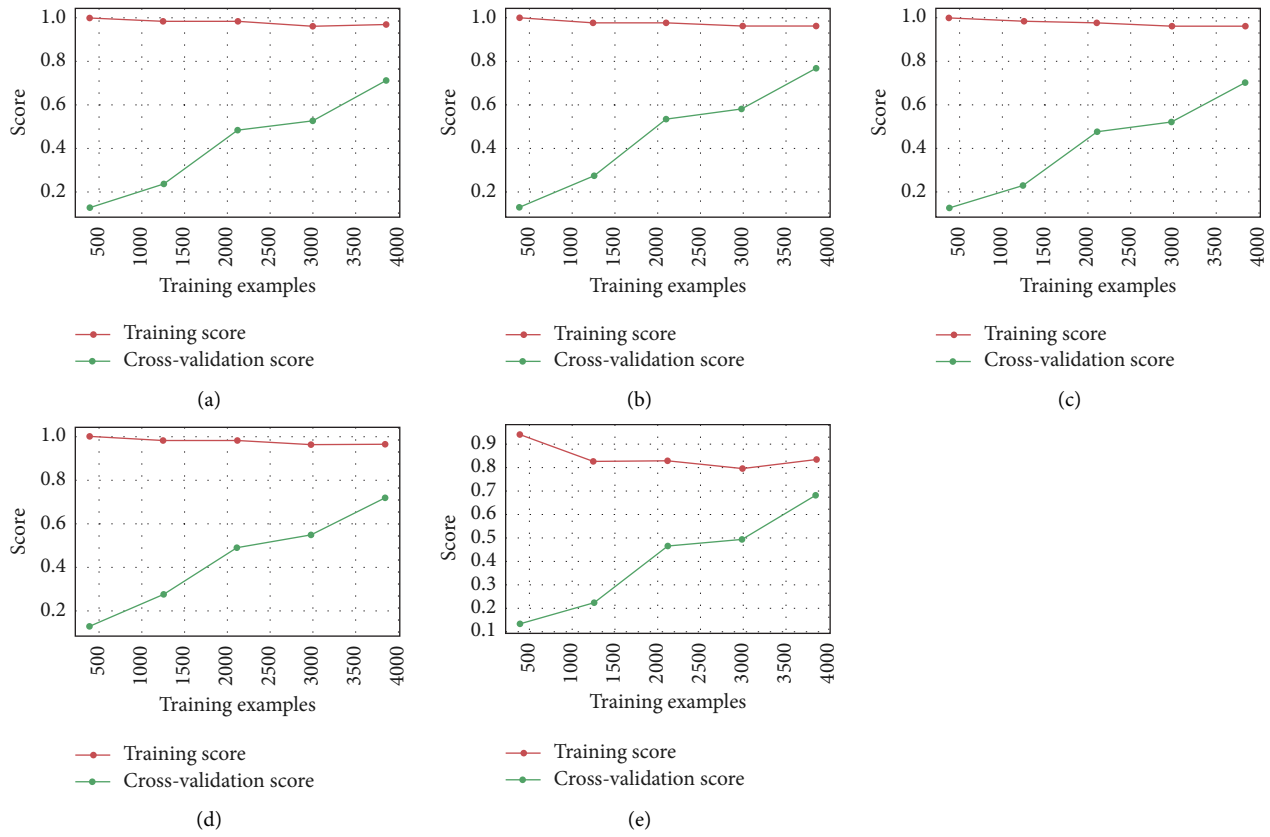

Figure 5: Learning curves of each method. (a) RF meaning curves. (b) SVC learning curves. (c) ExtraTrees learning curves. (d) AdaBoost learning curves. (e) GradientBoosting learning curves.

classifiers for ensemble modeling on the HUFF dataset, we also show the data distribution in Figure 4. We used learning curves to see the effect of overfitting the training set and the effect of the training size on the accuracy shown in Figure 5.

GradientBoosting and AdaBoost classifiers tend to overfit the training set. According to the growing cross-validation curves, GradientBoosting and AdaBoost could perform better with more training examples. The SVC and ExtraTrees classifiers seem to better generalize the prediction since the training and cross-validation curves are close together. By comparing the ensemble model with our DistilBERT BI-LSTM predictor, which is known as DBLP, the confusion matrix results are shown in Figure 6. DBLP improved in almost all categories and achieved a higher hit rate.

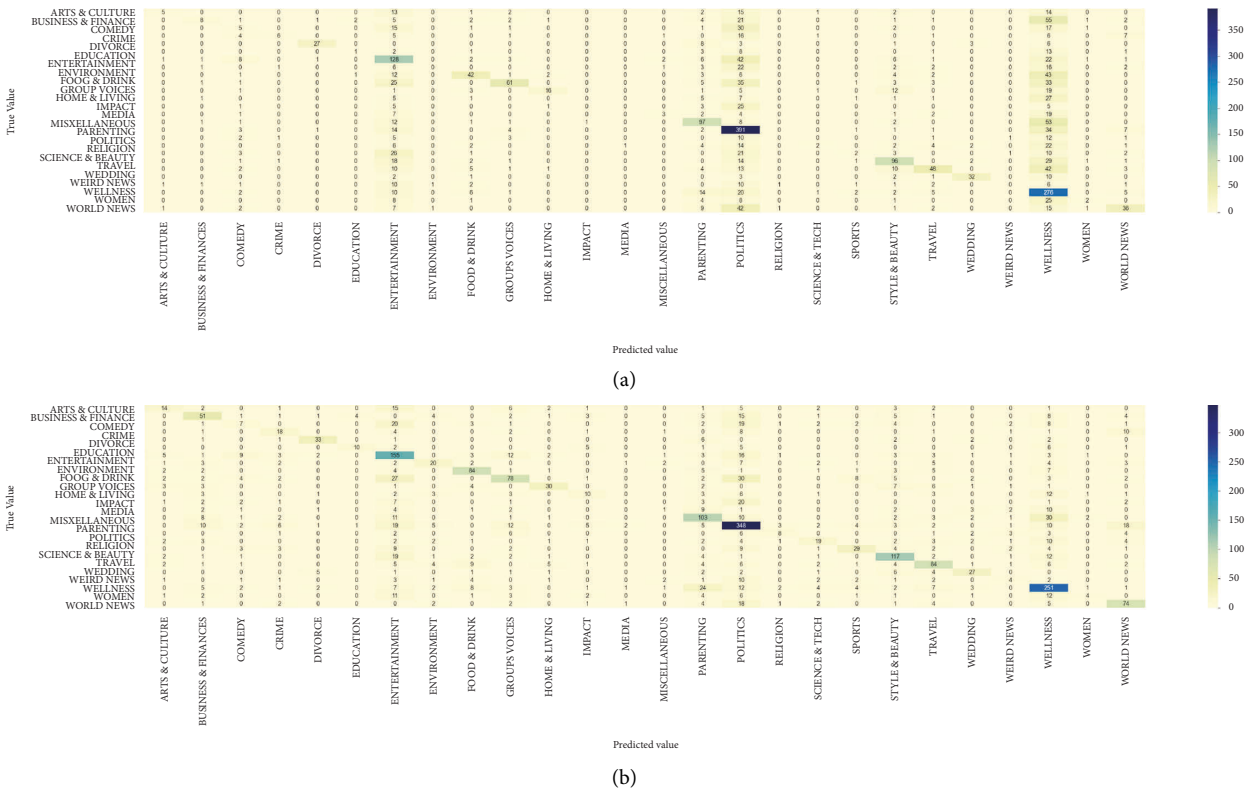

Figure 6: Confusion matrix comparison. (a) Ensemble modeling confusion matrix. (b) DBLP confusion matrix.

4.2.2. Comparison with Multimodel Methods. We conducted comparative experiments on the COVID-Q dataset. The composition of the data is shown in Figure 7, where text is the text content of the question and category represents the category of the question. We used a total of 4113 data points for training in 15 categories, as shown in Figure 8. The data for each category are shown in the figure below. It can be seen that most data have 280 data points, and the last data point has 230 data points. Our data are imbalanced. Regarding the test data, we used a total of 668 pieces of data, which is also imbalanced.

From Table 1, we can see that the precision of the DBLP model is 60% in 84 minutes, while the best accuracy is 61% based on the BERT + BI-LSTM model, but its training time is also the longest, and it takes 380 minutes. Moreover, we also see that the DistilBERT-CNN method achieves the fastest training speed, but at the same time, the accuracy is also reduced more. Our method obtains the highest F1 value. It shows that the bidirectional-layer LSTM network can better capture the fine-grained classification of semantic information between phrases and clauses in a small corpus. When DBLP is only 1% lower than the BERT-based method, the time is 100 minutes lower and only 84 minutes.

The total parameters of the model with DistilBERT decreased significantly, but the trainable parameters were only related to the feature extraction network. The model with the

	text	category
0	will covid stay and last perpetually	Speculation
1	will covid stay and last forever	Speculation
2	will covid stay forever last and	Speculation
3	will covid stay and forever	Speculation
4	will covid stay and last everlastingly	Speculation
...	...	...
4108	what should you do you suspect you have covid	Individual Response
4109	what should you do if suspect you you have covid	Individual Response
4110	what should you do if you defendant you have c...	Individual Response
4111	what should you if you suspect you have covid	Individual Response
4112	what should you do if birth you suspect you ha...	Individual Response

4113 rows × 2 columns

Figure 7: COVID-Q data composition.

CNN only achieved the smallest number of trainable params 94319, but its accuracy was also the lowest. This is because the CNN cannot well express the information of context and the problem of polysemy representation. On the number of training rounds, we set the early stopping function, which looks at the validation of the model loss score: if there is no improvement after 3 epochs, stop the training. Our model finally stops after 5 epochs. For the final model, because DistilBERT has only six layers of encoders, it is approximately half as small as the pretraining model based on BERT.

In summary, the advantages of our model are reflected in the following aspects. It requires fewer trainable parameters and saves computing resources. It can also be easily used by

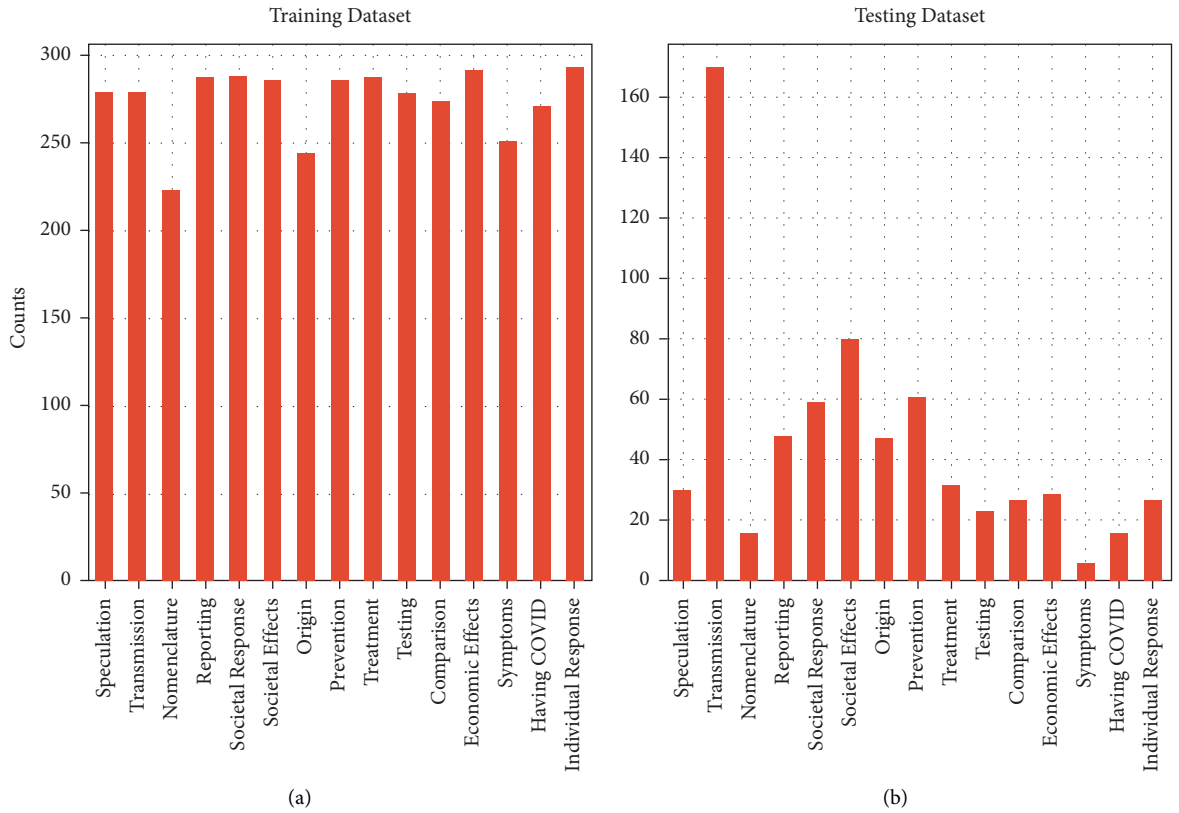

Figure 8: The distribution of COVID-Q data. (a) Training dataset. (b) Testing dataset.

Table 1: Method comparison.

Model	Precision (%)	Recall (%)	F1-score (%)	Time (min)	Total params	Trainable params	Epoch	Size (MB)
BERT + BI-GRU	60	54	54	241	109,920,145	1,609,873	6	419
BERT-BI-LSTM	**61**	53	54	380	109,247,003	936,731	11	417
BERT + CNN	58	51	52	153	108,404,591	94,319	4	413
BERT–CNN–LSTM	57	51	52	301	109,200,743	890,471	8	416
BERT-LSTM-CNN	52	44	44	189	111,918,864	3,608,592	5	427
DistilBERT + BI-GRU	59	54	54	85	66,800,785	1,609,873	5	261
DistilBERT BI-LSTM	**60**	**54**	**54**	**84**	**66,127,643**	**936,731**	**5**	**252**
DistilBERT-CNN	56	53	53	**69**	65,285,231	94,319	4	249
DistilBERT–CNN–LSTM	59	50	51	132	66,081,383	890,471	7	252
DistilBERT-LSTM-CNN	56	46	48	75	68,799,504	3,608,592	4	268

general in-depth learning enthusiasts and is conducive to deployment on the mobile terminal.

4.2.3. Performance of Real-World Data Sets. Finally, we crawled 4896 articles from the Mechatronics Encyclopedia website, with the abstract of the article as our training content and the topic of the article as the label to see how our model performed on the real-world task, with 27 categories in total, and the last category has only 49 data points. The distribution of the dataset is shown in Figure 9.

We divide the dataset into 64% train, 16% val, and 20% test. Thus, we use ranking metrics, such as accuracy and loss, to check our classifier. We compare the ranks produced by our classifier (ensemble classifier and DBLP classifier) to the SVM model, which is suitable for an extremely small dataset. The results are shown in Figure 10 and Table 2.

The result shows that the SVM classifier can solve the problem at some stage it has a small size and can be converged very quickly. However, SVM is limited to the recognition of the presence of words in the training dataset and is unable to capture the inner relationship of the sentence. In some sentences, the appearance of multiple keywords fooled the SVM model to label, for example, while the negation in the first place completely converted the case. The Ensemble model integrates vectors from multiple models so it has a big model size and slow training time. The DBLP can capture the semantic features of the text better than the SVM model and Ensemble model. Overall, the classification result was good, with accuracy of 86.65% and f1-score of 85%. By looking at some mislabeled data, we must admit that these texts are ambiguous and hard to classify.

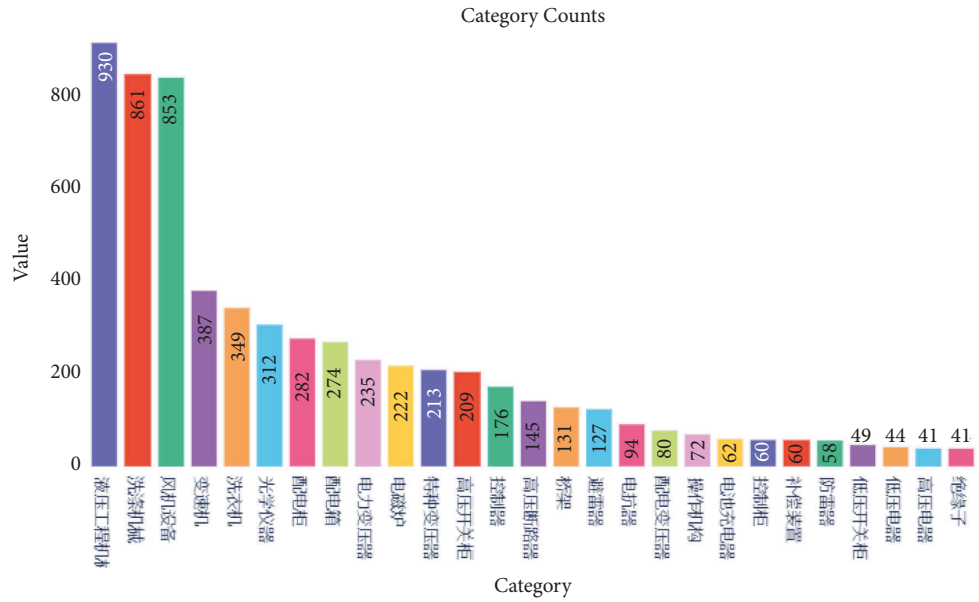

Figure 9: The distribution of the mechatronics encyclopedia dataset.

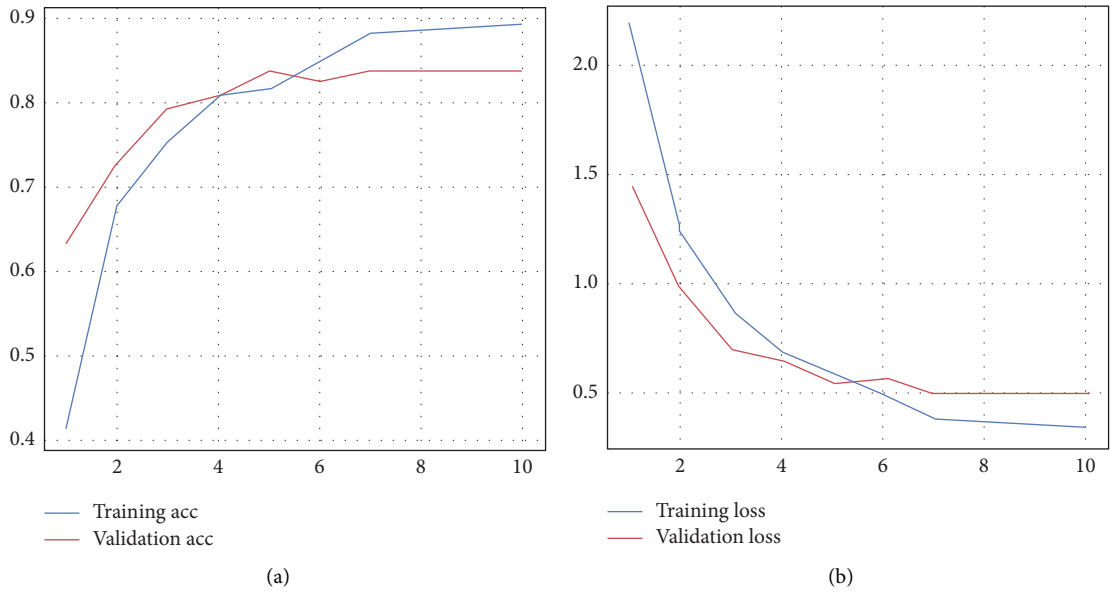

Figure 10: Training and validation accuracy in real-world data. (a) Training and validation accuracy. (b) Training and validation loss.

Table 2: Comparison of models for log case classification.

Classification task	Model	Accuracy (%)	F1-score (%)	Time (min)	Size (MB)
Log case classification (27 categories)	SVM	80.28	78	32	228
	Ensemble model	84.04	82	313	751
	DBLP	86.65	85	45	252

5. Conclusion

The results show that the DBLP model is guaranteed to be able to solve end-to-end data imbalance, small text, and multiclassification tasks and can be directly applied to real-world tasks. Furthermore, we can see that since we utilize the pretrained model after distillation, the training parameters of our model drop significantly, which is very friendly to general developers. Due to the smaller model size, our model can better apply AI technology to mobile smart devices, such as new energy vehicles and smart home appliances. In contrast, although the BERT-based pretraining model can achieve better accuracy, the computational cost and application cost are high, and it is more suitable for large-scale cloud computing and scientific research scenarios such as research institutes.

In future research, we will improve the model in the following two parts. We will try to combine semantic generation models to complement the important context that may be missing in sentences and apply our model to various tasks on smart devices.

Data Availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also form a part of an ongoing study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61876186) and the Xuzhou Science and Technology Project (No. KC21300).

References

[1] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: a survey," *Information*, vol. 10, no. 4, p. 150, 2019.

[2] X. Li, "Chinese language and literature online resource classification algorithm based on improved SVM," *Scientific Programming*, vol. 2022, Article ID 4373548, 7 pages, 2022.

[3] K. Vaish, G. Deepak, and A. Santhanavijayan, *DSEORA: Integration of Deep Learning and Metaheuristics for Web Page Recommendation Based on Search Engine Optimization Ranking Emerging Research in Computing, Information, Communication, and Applications*, pp. 873–883, Springer, Singapore, 2022.

[4] Y. Chen, D. Liu, Y. Liu, Y. Zheng, B. Wang, and Y. Zhou, "Research on user-generated content in Q&A system and online comments based on text mining," *Alexandria Engineering Journal*, vol. 61, no. 10, pp. 7659–7668, 2022.

[5] H. Lou, H. Zhao, and W. Deng, "Research on civil aviation passenger question intention recognition based on text classification method of self-attention and deep neural network," in *Proceedings of the 2021 Chinese Intelligent Automation Conference*, pp. 286–293, Springer, Zhanjiang, China, November 2021.

[6] H. Chen, L. Wu, J. Chen, W. Lu, and J. Ding, "A comparative study of automated legal text classification using random forests and deep learning," *Information Processing & Management*, vol. 59, no. 2, Article ID 102798, 2022.

[7] A. N. Tarekegn, M. Giacobini, and K. Michalak, "A review of methods for imbalanced multi-label classification," *Pattern Recognition*, vol. 118, Article ID 107965, 2021.

[8] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: pre-training of deep bidirectional transformers for language understanding," 2018, https://arxiv.org/abs/1810.04805.

[9] Y. Song, J. Wang, Z. Liang, Z. Liu, and T. Jiang, "Utilizing BERT intermediate layers for aspect-based sentiment analysis and natural language inference," 2020, https://arxiv.org/abs/2002.04815.

[10] C. X. Wan and B. Li, "Financial causal sentence recognition based on BERT-CNN text classification," *The Journal of Supercomputing*, vol. 78, no. 5, pp. 6503–6527, 2022.

[11] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper, and lighter," 2019, https://arxiv.org/abs/1910.01108.

[12] G. Xiong and K. Yan, "Multitask sentiment classification model based on DistilBERT and multiscale CNN," in *Proceedings of the 2021 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 700–707, IEEE, AB, Canada, October 2021.

[13] H. Adel, A. Dahou, A. Mabrouk et al., "Improving crisis events detection using DistilBERT with hunger games search algorithm," *Mathematics*, vol. 10, no. 3, p. 447, 2022.

[14] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, https://arxiv.org/abs/1503.02531.

[15] T. H. Hsu, Z. H. Wang, and A. R. See, "A cloud-edge-smart IoT architecture for speeding up the deployment of neural network models with transfer learning techniques," *Electronics*, vol. 11, no. 14, p. 2255, 2022.

[16] J. Glory Precious, S. P. Angeline Kirubha, and I. Keren Evangeline, "Deployment of a mobile application using a novel deep neural network and advanced pre-trained models for the identification of brain tumours," *IETE Journal of Research*, pp. 1–13, 2022.

[17] Y. Yao, Y. Li, C. Wang et al., "Int8 winograd acceleration for conv1d equipped asr models deployed on mobile devices," 2020, https://arxiv.org/abs/2010.14841.

[18] M. Ghiassi, S. Lee, and S. R. Gaikwad, "Sentiment analysis and spam filtering using the YAC2 clustering algorithm with transferability," *Computers & Industrial Engineering*, vol. 165, Article ID 107959, 2022.

[19] Y. Kim, S. Bang, J. Sohn, and H. Kim, "Question answering method for infrastructure damage information retrieval from textual data using bidirectional encoder representations from transformers," *Automation in Construction*, vol. 134, Article ID 104061, 2022.

[20] O. Stitini, S. Kaloun, and O. Bencharef, "Integrating contextual information into multi-class classification to improve the context-aware recommendation," *Procedia Computer Science*, vol. 198, pp. 311–316, 2022.

[21] D. U. Yang, B. Kim, S. H. Lee, Y. H. Ahn, and H. Y. Kim, "AutoDefect: defect text classification in residential buildings using a multi-task channel attention network," *Sustainable Cities and Society*, vol. 80, Article ID 103803, 2022.

[22] Y. Ma, X. Liu, L. Zhao, Y. Liang, P. Zhang, and B. Jin, "Hybrid embedding-based text representation for hierarchical multi-label text classification," *Expert Systems with Applications*, vol. 187, Article ID 115905, 2022.

[23] S. Fakhar Bilal, A. Ali Almazroi, S. Bashir, F. Hassan Khan, and A. Ali Almazroi, "An ensemble based approach using a combination of clustering and classification algorithms to enhance customer churn prediction in telecom industry," *PeerJ Computer Science*, vol. 8, p. e854, 2022.

[24] K. Wang, Y. Liu, B. Cao, and J. Fan, "TkTC: a framework for top-k text classification of multimedia computing in wireless networks," *Wireless Networks*, pp. 1–12, 2022.

[25] Y. Zhu, X. Zhou, J. Qiang, Y. Li, Y. Yuan, and X. Wu, "Prompt-learning for short text classification," 2022, https://arxiv.org/abs/2202.11345.

[26] J. S. Andersen and W. Maalej, "Efficient, uncertainty-based moderation of neural networks text classifiers," 2022, https://arxiv.org/abs/2204.01334.

[27] J. W. Chang, N. Yen, and J. C. Hung, "Design of a NLP-empowered finance fraud awareness model: the anti-fraud chatbot for fraud detection and fraud classification as an instance," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 10, pp. 4663–4679, 2022.

[28] J. Wei, C. Huang, S. Vosoughi, and J. Wei, "What are people asking about covid-19? a question classification dataset," 2020, https://arxiv.org/abs/2005.12522.

[29] O. Sagi and L. Rokach, "Ensemble learning: a survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[30] R. Qasim, W. H. Bangyal, M. A. Alqarni, and A. Ali Almazroi, "A fine-tuned BERT-based transfer learning approach for text classification," *Journal of healthcare engineering*, vol. 202217 pages, Article ID 3498123, 2022.