Robust Text Classification via Improved CNN, Unbalanced BiLSTM, and Multi-Head Attention

Yan Liang*, Jiabin Liu
Department of Literature, Sichuan Minzu College, Kangding, 626000, China
E-mail: yanliangylyl2025@outlook.com
*Corresponding author

Keywords: natural language, text classification, BiLSTM, attention mechanism, feature fusion

Received: August 25, 2025

As one of the core tasks of natural language processing technology, text classification methods generally face the problems of insufficient global semantic capture and limited feature focusing ability when processing long texts or complex semantics. To address this issue, a deep learning model that integrates improved convolutional neural networks, unbalanced bidirectional long short-term memory networks, and multi-head attention mechanisms is proposed. Utilizing an improved bidirectional long short-term memory network to capture global semantic information, while dynamically focusing on key features through a multi head attention mechanism to enhance the model's adaptability to classification tasks. The performance of the model is validated through experiments on AG News (short text) and IMDb (long text) datasets. The results show that in short text classification, the proposed method has an accuracy rate of 96% and a classification error rate of only 1.46%. In the task of long text classification, the method proposed in the study has a product under the curve of 0.98. In adversarial attack testing, the accuracy rates of adversarial samples generated by different methods are 92.85% and 90.63%, respectively, with the lowest robustness degradation rates of 3.72% and 5.49%, respectively. In cross domain generalization testing, it shows the least classification errors and superior cross domain adaptability. These results validate the high performance, robustness, and wide applicability of the method. The research indicates that this approach can validly improve the performance of text classification and provide new solutions for natural language processing related tasks in long text and multi-category scenarios.

Povzetek:

1 Introduction

With the rapid advancement of information technology, Natural Language Processing (NLP), as an important branch of AI, is commonly applied in various scenarios such astext categorization, emotional analysis, data retrieval, and automated language translation [1-2]. In these applications, text classification (TC), as one of the core tasks, plays an important role in transforming massive unstructured text data into structured information. In recent times, with the explosive growth of the amount of information on the Internet, how to accurately and efficiently classify a large number of texts has garnered significant attention as the focal point of common concern in academia and industry. Traditional TC methods mainly rely on manually designed features and shallow machine learning models, such as Naive Bayes, Support Vector Machines, etc. [3]. Although these methods have shown certain effectiveness in specific tasks, they have many limitations in complex scenarios due to the difficulty in capturing deep semantic features of text. To address these challenges, deep learning models have increasingly emerged as the predominant approach in the domain of TC due to their powerful feature extraction capabilities [4]. Among them, Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) have been widely used due to their superior processing capabilities for time series and local features [5]. When using the LSTM model alone, its feature extraction ability is easily limited to the directionality of the time series, resulting in the inability to fully utilize the bidirectional contextual information of the text. Secondly, CNN models perform well in extracting local features, but their ability to capture global semantics is weak [6]. In addition, existing methods are often susceptible to the interference of redundant information when facing long texts, making it difficult to effectively focus on key semantic information, thereby reducing classification accuracy. In view of this, a TC method combining Bidirectional Long Short-Term Memory Network (BiLSTM) and attention mechanism (AM) is proposed. BiLSTM is utilized to select global contextual information, and AM is combined to focus on key features, achieving effective fusion of global and local features and improving classification performance. The research aims to improve the capability of TC tasks, especially in long text and multi-category scenarios, through the proposed method, providing new solutions for NLP related tasks. The study aims to verify two hypotheses. Firstly, compared to the standard BiLSTM symmetric processing of context, the unbalanced BiLSTM (UBBiLSTM) can effectively model context representation through asymmetric weighting mechanism, thereby improving classification performance. Secondly, by integrating multi-head AMs, the model can more accurately focus on key semantic information in the text, thereby improving its robustness and generalization ability in conventional tasks, adversarial attacks, and cross domain scenarios.

2 Related works

Natural language TC is a crucial task in NLP, widely used in scenarios such as emotion assessment, junk mail identification, and categorization of news content. With the development of machine learning and deep learning, TC approaches have gradually evolved from traditional statistical models to efficient models based on deep learning, achieving good results. Mohammed A et al. proposed a new meta learning ensemble method to address the problem of selecting the most suitable deep learning classifier for TC tasks. By using a two-level meta classifier to fuse with a baseline deep learning model, the classification accuracy of the baseline deep model was improved, and the performance exceeded that of the current leading ensemble techniques [7]. Soni S proposed a novel architecture TextConvoNet based on CNNs for TC problems. It not only identified n-gram features within individual sentences but also detected n-gram patterns across sentences within the input text data. By using two-dimensional multi-scale convolution operations, the performance of TC was improved [8]. Jalal N et al. proposed an improved random forest model for TC, called Improved Random Forest TC. This model combined self sampling and random subspace methods. Its purpose was to optimize the performance of traditional random forests and other machine learning models by removing unimportant features, increasing the number of trees in the forest, and monitoring the classification capability of random forests [9]. Garrido Merchan E C et al. proposed a comparison of the performance of the BERT model and the traditional word frequency inverse document frequency model when input into a machine learning model to improve TC performance in NLP tasks. Through a series of empirical tests in different scenarios, they demonstrated the superiority of the BERT model and its universality without being affected by text language features [10].

To overcome the pain of heterogeneous data, black box unresolvability, and cross domain migration, Yao J et al. proposed a regular optimization system consisting of multi-head sparse attention, adaptive Focal Loss, LayerNorm preheating, and AdamW decay. The results showed that in TC, entity recognition, and reading comprehension, the Macro-F1 of this method improved by an average of 4.6%, and the edge inference delay was reduced to 1/3 [11]. Umer M et al. proposed a FastText initialization embedding and joint fine-tuning scheme to address the challenge of insufficient word vector representation in CNN TC performance. The results showed that Macro-F1 improved by an average of 3.2%, verifying the effectiveness of the strategy [12]. Kenarang et al. proposed an approach that combined bidirectional gated recurrent units, AMs, and capsule networks to address the issue of topic recognition in news classification. This improved the capability of Persian TC and solved the problem of relevance of important vocabulary in long texts [13]. Enamoto et al. proposed a BiLSTM network model to address the difficulty of information extraction caused by the complexity of legal texts. By combining attention layers, the performance of Portuguese legal TC was optimized, achieving the capture of past and future contexts of long judicial texts and fast processing of multi-label and multi-class datasets [14].

In summary, existing research has made progress in TC performance and adaptability, but there are still problems such as insufficient global semantic capture of long texts and unstable performance in low resource scenarios. Meanwhile, although there have been many models combining BiLSTM with Attention, these methods can easily lead to the loss of local key features and a single dimension of AM when dealing with complex text tasks. In recent years, many deep learning-based TC fusion models have emerged, but these methods also have many shortcomings, as shown in Table 1. The innovation of the research lies in the introduction of an improved CNN module to enhance the extraction of local n-gram features. Meanwhile, UBBiLSTM is proposed, which utilizes an asymmetric weight fusion mechanism to more flexibly adjust the importance of forward and backward contexts compared to traditional BiLSTM. In addition, the study systematically combines the above structure with multi-head AM, while improving the model's feature expression, robustness, and generalization ability, not just accuracy. This is in stark contrast to most models that simply concatenate CNN+LSTM or BiLSTM+Attention.

Table 1: Summary of TC related research

Research author	Model name	Core component	Key performance indicators	Limitation
Mohammed A et al. [7]	Meta learning ensemble model	Two level meta classifier+baseline DL model	2% -3% better than existing integration methods	Large number of parameters
Soni S[8]	TextConvoNet	Two-dimensional multi-scale CNN	Improve the accuracy of local feature extraction by 4%	Weak ability to capture global semantics
Jalal N et al. [9]	Improved Random Forest	Self sampling+random subspace	5% increase compared to traditional random forest F1	Unable to capture deep semantic features
Garrido Merchan E C et al. [10]	BERT+ML model	Pre trained BERT+TF-IDF	Improved accuracy by 8% - 10% compared to traditional ML models	High computational cost during long text processing
Kenarang et al. [13]	BiGRU+Attention+ Capsule Network	Bidirectional GRU+Single Head Attention	Accuracy rate 91.2%	Not adapted to multiple languages, robustness not verified

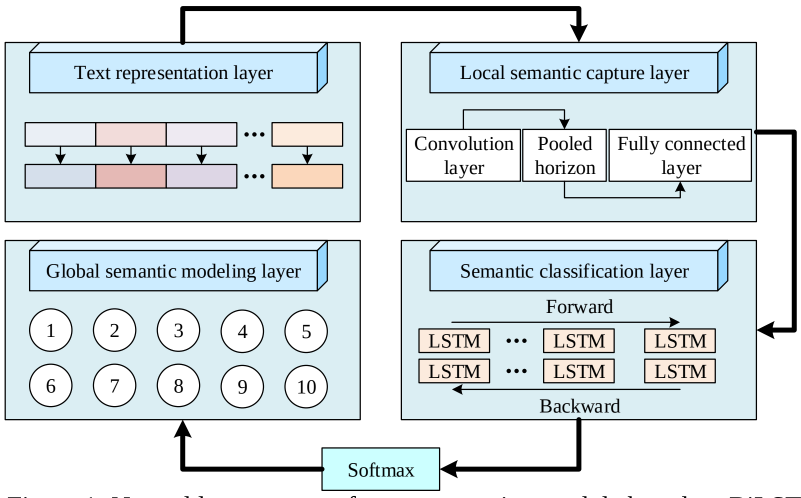
Enamoto et al. [14]	BiLSTM+Attention	Standard BiLSTM+Single Head Attention	Multi label classification F1 89.5%	Significant interference caused by long text redundancy
This study	Improved CNN+UBBiLSTM+ MHA	Multi scale CNN+Unbalanced BiLSTM+Multi head Attention	AG News has an accuracy rate of 96% IMDb AUC 0.98	/

3 Methods and materials

This section mainly explains the natural language TC method grounded on BiLSTM and AM. Specifically, BiLSTM is first used to extract local and global (L&G) contextual features of the text, followed by the fusion of AMs to concentrate on essential characteristic data and optimize feature representation. Ultimately achieving efficient feature fusion and improving the performance of TC.

3.1 Natural language text feature extraction based on BiLSTM

A classification method that integrates BiLSTM and AM is proposed to address the problem of difficult extraction of deep semantic features in natural language TC. As an important task in NLP, the key to natural language TC lies in accurately extracting semantic information from the text [15]. However, traditional methods often face the problems of insufficient local feature expression and limited global semantic association modeling ability when dealing with long texts or complex semantics [16]. Therefore, the proposed classification method extracts L&G features of the text through BiLSTM, and combines AM to concentrate on essential data related to the classification task, thereby achieving effective fusion of L&G features. Among them, the natural language text feature extraction module based on BiLSTM is the core part of the method, and its structure is shown in Figure 1.


Text representation layer
Local semantic capture layer
Convolution layer
Pooled horizon
Fully connected layer
Global semantic modeling layer
1
2
3
4
5
6
7
8
9
10
Semantic classification layer
Forward
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
Backward
Softmax
Figure 1: Natural language text feature extraction module based on BiLSTM

As shown in Figure 1, the natural language text feature extraction module consists of a text representation layer, a local semantic capture layer, a global semantic modeling layer, and a semantic classification layer. Among them, the text representation layer converts the original text into a word vector matrix for subsequent layers to process. The local semantic capture layer utilizes an improved CNN to extract local features through multi-scale convolution, and optimizes the representation of local features through pooling and fully connected operations. The global semantic modeling layer adopts a network, combined with a forward backward asymmetric weighting mechanism, to capture the global semantic information of the text context. The semantic classification layer receives features and performs softmax classification to output the final text category. The specific workflow of the local semantic capture layer in the module is shown in Figure 2.

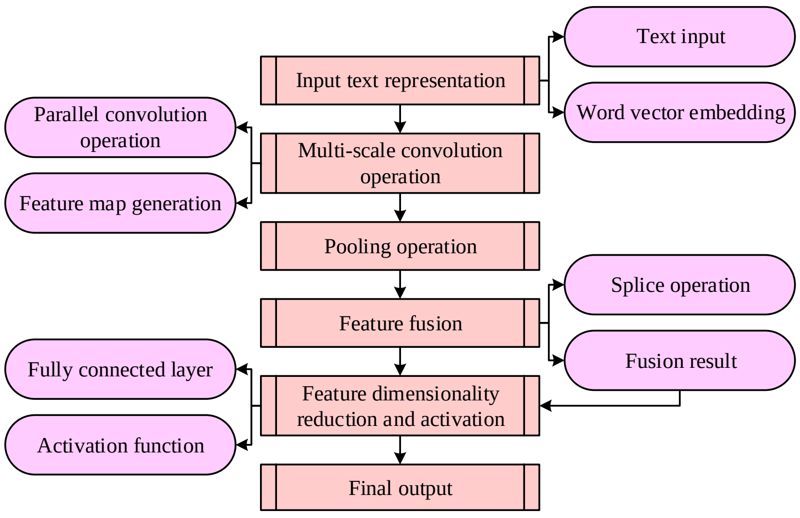


Figure 2: Specific workflow of the local semantic capture layer

As shown in Figure 2, the core of the local semantic capture layer is to use an improved CNN to extract multi-scale local features from the input text, and optimize feature representation through pooling and dimensionality reduction operations. In this layer, the text is first transformed into a matrix representation of word vectors. Assuming the input text contains n words with a word vector dimension of d, the input matrix X is represented as equation (1).

$$X = \left[\omega_1, \omega_2, ..., \omega_n\right] \in \mathbb{R}^{n \times d} \quad (1)$$

In equation (1), ω_i is the word vector of the i th word. To extract local features, convolution operations use multiple convolution kernels (CKs) of different sizes to slide the input matrix X. Assuming the size of the CK is $h \times d$, the output of the convolution operation is represented by equation (2).

$$c_i = f\left(K \cdot X_{i:i+h-1} + b\right) \quad (2)$$

In equation (2), c_i represents the eigenvalue of a local window in the input, $X_{i:i+h-1}$ represents the submatrix composed of the i th word to the $i+h-1$ th word, K represents the weight matrix of the CK, b is the bias term, and $f(\cdot)$ is the activation function. By sliding the CK on the input matrix, a one-dimensional feature map can be generated, as shown in equation (3).

$$C = \left[c_1, c_2, ..., c_{n+h-1}\right] \in \mathbb{R}^{n+h-1} \quad (3)$$

To capture n-gram features of different lengths, the local semantic capture layer uses multiple sizes of CKs for parallel operations, and each CK generates a feature map. After the convolution operation, to reduce redundant features and improve the robustness of the model, the local semantic capture layer uses max pooling operation to process each feature map, as shown in equation (4).

$$c_i^{(k)} = f\left(\sum_{u=1}^{h}\sum_{v=1}^{d} K_{u,v}^{(k)} X_{i+u-1,v} + b^{(k)}\right) \quad (4)$$

In equation (4), h and d respectively represent the height and width of the CK, as well as the number of covered words and the dimension of word vectors. $K_{u,v}^{(k)}$ represents the u th and v th elements of the CK, $X_{i+u-1,v}$ is the elements in row $i+u-1$ and column v of the input matrix, $b^{(k)}$ is the bias of the CK, and $f(\cdot)$ is the activation function. After the convolution operation, to reduce redundant features and improve the robustness of the model, max pooling operation is used to process each feature map. The result of pooling operation compresses each feature map into a fixed length feature vector. For pooled feature vectors generated by multiple CKs, they can be combined into a unified feature representation through concatenation operations [17]. The fused feature vector undergoes dimensionality reduction through a fully connected layer and introduces nonlinearity through an activation function. In the global semantic modeling layer, the use of Unbalanced BiLSTM (UBBiLSTM) is studied to solve the problem of traditional BiLSTM simply concatenating forward and backward hidden states (HSs) into comprehensive features, which cannot flexibly adjust the importance of forward and backward features. UBBiLSTM is an improvement of the standard BiLSTM in this study. In natural language, the contribution of forward and backward context to semantics is often asymmetric. The standard BiLSTM fuses forward and backward information equally by simple concatenation or addition, ignoring this difference. UBBiLSTM introduces a learnable parameter to weight and fuse the forward HS and backward HS. This parameter is optimized end-to-end through backpropagation during model training, enabling it to automatically learn the optimal fusion ratio of forward and backward context based on task data. Compared with existing asymmetric fusion methods, UBBiLSTM has significant differences. The gate control mechanism of BiRNN requires the design of independent gate units (and the introduction of additional parameters to calculate gate weights), which can achieve asymmetric fusion, but has high structural complexity and computational cost. However, UBBiLSTM simplifies the fusion logic through a single learnable parameter, ensuring the ability to capture asymmetric semantics while only increasing the number of parameters slightly, effectively balancing performance improvement and computational efficiency. The network structure of UBBiLSTM is in Figure 3.

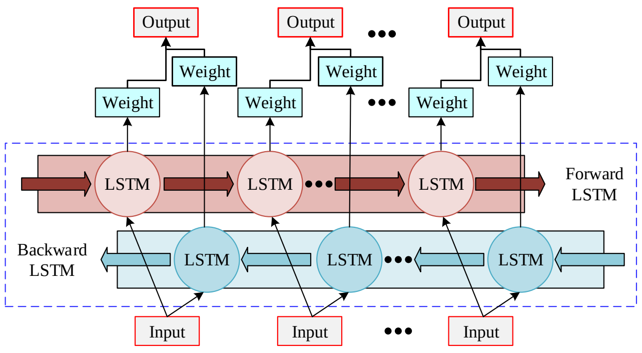


Figure 3: Schematic diagram of UBBiLSTMs network structure

As shown in Figure 3, UBBiLSTM consists of an input layer, a BiLSTM layer, an asymmetric weight fusion layer, and an output layer. Firstly, the input text is transformed into a sequence of word vectors, which are then simultaneously fed into both forward and backward LSTM networks. Forward LSTM gradually processes text from left to right, capturing semantic information of the current word and subsequent context. Backward LSTM processes text from right to left, capturing semantic information of the current word and its preceding context. In traditional BiLSTM, the semantic information of forward and backward is directly concatenated together to form the output, while the core of UBBiLSTM is to use an asymmetric weight fusion mechanism to weight and combine the HSs of forward and backward LSTM, thereby generating a comprehensive HS, as shown in equation (5).

$$h_t = \beta \cdot \overrightarrow{h_t} + (1-\beta) \cdot \overleftarrow{h_t} \tag{5}$$

In equation (5), h_t is the comprehensive HS of the time step t, $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are the HSs of the forward and backward LSTM in the time step t, and β represents the weights that control the importance of forward semantics. UBBiLSTM mainly achieves asymmetric modeling through independent gating parameters combined with dynamic weighting factors. Forward and backward LSTM use independent gating weights to learn the semantic rules of "front text → current" and "back text → current" respectively, while introducing learnable β to dynamically adjust the fusion ratio of the HSs in the front and back directions. During training, β is initialized to 0.5, and the loss gradient will guide β and gate parameters to adaptively adjust based on the difference in forward and backward semantic contributions, thereby achieving end-to-end learning of asymmetric fusion.

3.2 TC Optimization design incorporating AM

In the BiLSTM-based natural language text feature extraction module, the combination of local semantic capture layer and global semantic modeling layer can effectively model the local features and global contextual information of the text. However, relying solely on BiLSTM to extract features may not fully distinguish between task related important information and irrelevant redundant information, which may limit the generalization ability of feature expression [18]. Therefore, the study introduces AMs to further optimize the features extracted by BiLSTM, focusing on key information highly relevant to the classification task, thereby improving the capability and accuracy of TC. The overall framework of the TC method that integrates AM is in Figure 4.

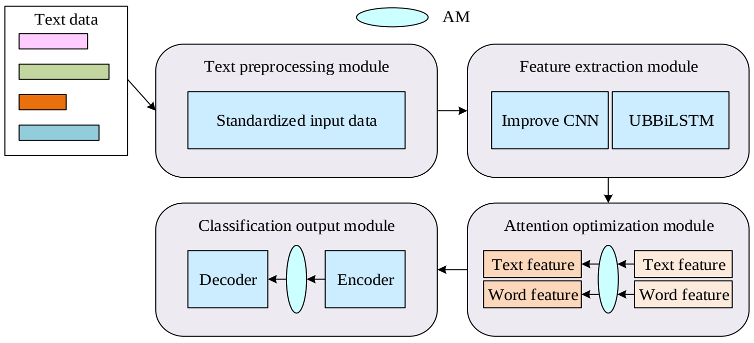


Figure 4: Optimal design of TC with AM

As presented in Figure 4, the general structure of the TC method after integrating AM includes four main modules, namely text preprocessing module, feature extraction module, attention optimization module, and classification output module. The text preprocessing module has a similar function to the text representation layer in the BiLSTM-based natural language text feature extraction module, mainly used to generate standardized input data. The feature extraction module, as mentioned earlier, consists of an improved CNN and UBBiLSTM. On this basis, the attention optimization module fuses and weights the features generated by the feature extraction module. By dynamically assigning feature weights, it can effectively focus on key features related to classification tasks and weaken the influence of redundant features. The AM can not only highlight important semantic regions in text, but also improve the capability of the model in long texts or complex semantic scenes. Finally, the attention optimized features are input into the decoder for further feature integration, and the final category of the text is output through the Softmax classifier.

The attention optimization module, as the core link between feature extraction and classification output, not only improves the quality of feature representation, but also provides more accurate feature support for classification tasks. The research introduces a multi-head AM, whose core idea is to model input features in parallel through multiple independent attention heads, with each head focusing on capturing semantic information from specific dimensions or contexts. The specific structure is in Figure 5.

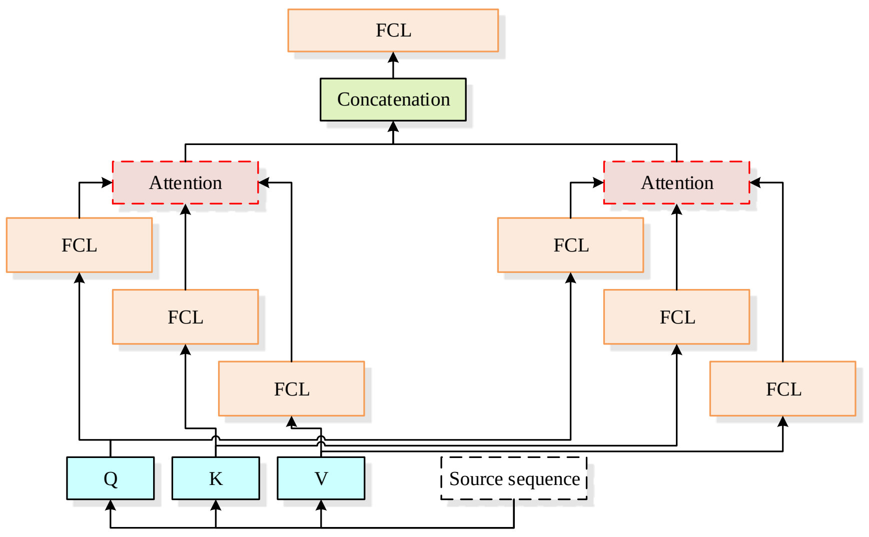


Figure 5: Structure diagram of multi-head AM

As shown in Figure 5, in the multi-head AM, the input text sequence is first mapped to query, key, and value vectors, and the generation of vectors is completed through the learned linear transformation matrix. In TC tasks, the query vector represents the semantic features that the classification model wants to focus on, the key vector represents the identification of each word in the text sequence, and the value vector represents the specific content of each word, namely the word embedding, used to generate contextual feature representations related to the classification task. Furthermore, by determining the degree of resemblance between queries and keys through dot product, attention weights are generated, and then combined with value vectors for weighted summation to capture key information in the text that is highly relevant to the classification target. The multi-head AM utilizes weight allocation to capture the relationships between features, as shown in equation (6).

$$\text{Attention}(Q,K,V)=\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

In equation (6), Q, K, and V are the query, key, and value vectors. d_k indicates the dimension of the key vector. In TC tasks, multi-head AM is used to calculate the correlation between each word in the input text and other words, to dynamically adjust the attention focus of the model. For example, in sentiment classification, AMs focus on keywords that contain emotional information while weakening the influence of irrelevant information. When multiple attention heads are parallel, different attention heads can capture different features of the text, such as one head focusing on emotional vocabulary and the other head capturing contextual relationships. The output context vectors of each attention head are concatenated and integrated to generate an overall feature representation containing multi-level semantic information, thereby improving the performance and accuracy of TC. To fully utilize the global contextual features of UBBiLSTM and the key semantic features of MHA, an ensemble approach of residual connections and layer normalization is adopted instead of simple concatenation or addition. In addition, the study incorporates a decoder-encoder design into the TC framework and also introduces AMs, as shown in Figure 6.

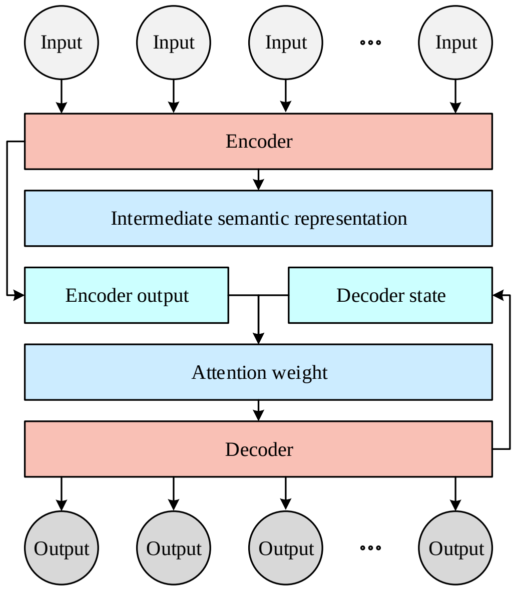


Figure 6: Encoder-decoder structure with AM

As presented in Figure 6, the overall structure includes an encoder and a decoder, where the encoder receives an input text sequence, generates a set of intermediate semantic representations, and captures global semantic information. The decoder utilizes the output of the encoder and dynamically generates the target output based on the current decoding state. After introducing AM, the decoder can assign weights to each output of the encoder and dynamically adjust the attention to different input parts. At each decoding moment t, the AM calculates attention weight $\alpha_{t,i}$ based on the current decoder state s_t and encoder output C_i, as shown in equation (7).

$$\begin{cases} \alpha_{t,i} = \dfrac{\exp\left(e_{t,i}\right)}{\sum_{j=1}^{n} \exp\left(e_{t,i}\right)} \\ e_{t,i} = \tanh\left(W_q s_t + W_k C_i + b_e\right) \end{cases} \tag{7}$$

In equation (7), $e_{t,i}$ represents the correlation between the decoder state and the encoder output. W_q and W_k represent the weight matrices of the query vector and key vector, respectively, and b_e is the bias term. Subsequently, the generated attention weights are used to weight the output of the summing encoder, generating a context vector as shown in equation (8).

$$C_t = \sum_{i=1}^{n} \alpha_{t,i} C_i \tag{8}$$

Through the above calculation steps, the AM can dynamically focus on the most important parts of the input text for classification tasks and adjust the focus points at different decoding times. Meanwhile, the decoder combines the joint information of context vectors and HSs to generate more accurate classification results. In this model, the query, key, and value vectors input to the MHA mechanism are all from the output HS sequence of UBBiLSTM. Due to the fact that UBBiLSTM has already processed the text in order and its output already contains positional information, the study did not add additional absolute positional encoding to avoid information redundancy. The output of the MHA module is a weighted sequence of context vectors. This sequence is integrated with the original output sequence of UBBiLSTM through concatenation, and then dimensionality reduction and feature fusion are performed through a fully connected layer, and finally input to the decoder or classification layer.

The research model training is divided into four steps. Firstly, the data preprocessing stage uses the NLTK toolkit to clean the original text. The GloVe-300d pre trained word vector is used to initialize the word embedding layer, and the text sequence is uniformly truncated/padded to a fixed length of 512. Next, the training enters the model initialization phase to initialize the parameters of each module. Improved CNN adopts three sizes of convolution kernels [3,4,5], each with 128 kernels. The hidden layer dimension of UBBiLSTM is 256, with 8 multi-head attention heads and 64 dimensions per head. Subsequently, during the model training phase, the AdamW optimizer (learning rate of 0.001, weight decay of 0.01) is used to minimize cross entropy loss, and the Cosine Annealing learning rate scheduler is used to dynamically adjust the learning rate. The batch size is set to 64, and the total number of training epochs is 30. Additionally, an early stop mechanism is introduced to terminate the training if there is no improvement in the F1 score of the validation set for 5 consecutive rounds, and the model weights with the best performance in the validation set are saved. Finally, in the model inference stage, the optimal model saved during training is loaded, and the test set text is preprocessed in the same way as in the training stage. After inputting the model, the probabilities of each category are output through the Softmax function, and the category corresponding to the highest probability is taken as the final prediction result.

4 Results

To confirm the validity and superiority of the proposed natural language TC method that integrates BiLSTM and AM, the study conducted a large number of experiments on a server configured with Intel Xeon E5-2698 v4 @ 2.20GHz (16 cores) CPU and NVIDIA Tesla V100 (16GB HBM2) GPU. The model training and evaluation were based on the PyTorch 1.12.0 framework, using the Python 3.8.13 environment, combined with tools such as Transformers, Scikit learn, NumPy, Matplotlib, and Pandas. All experimental datasets were divided into training set: validation set: test set=8:1:1, and an additional 5-fold cross validation was performed. The mean and standard deviation were taken as the final results to verify the statistical robustness of the model performance and avoid overfitting. The parameter settings required for the experiment are presented in Table 2.

Table 2: Configuration of experiment parameters

Category	Parameter	Value	Category	Parameter	Value
General settings	Optimizer	AdamW	CNN settings	Kernel sizes	[3, 4, 5]
	Learning rate	0.001		Number of kernels	128
	Weight decay	0.01		Activation function	ReLU
	Learning rate scheduler	Cosine annealing scheduler		Dropout rate	0.5
	Batch size	64		Max pooling window size	2
	Epochs	30	Multi-head attention (MHA) settings	Number of attention heads	8
	Random seed	42		Dimension per head	64
BiLSTM settings	Hidden layer size	256		Dropout rate	0.1
	Bidirectional	Yes	Decoder settings	Hidden layer size	128
	Dropout rate	0.5		Activation function	Tanh
/	/	/		Output layer	Softmax

Based on Table 2, the study selected the AG News dataset (URL: https://www.di.unipi.it/ ~Gulli/AG-corpus_of_news_articles. html) and IMDb dataset (URL: https://www.imdb.com/) as a source of experimental data. Among them, AG News is a widely-used news classification dataset used to test the performance of models in multi-category short TC tasks. IMDb is a well-known movie review classification dataset used to test the performance of models in binary long text sentiment analysis tasks. The study first conducted ablation experiments, and the outcomes are in Table 3.

Table 3: Outcomes of the ablation study

Model Variants	Accuracy (%)	F1 score (%)	Precision (%)	Recall (%)	Inference time (ms)	Number of parameters (M)
Without AM	91.2	89.7	91.5	88.2	21.3	13.8
Replace UBBiLSTM with BiLSTM	89.7	87.9	90.2	86.0	22.5	13.5
Without CNN	88.3	86.8	88.1	85.7	18.7	11.9
Replace MHA	86.5	85.3	86.2	84.7	24.1	14.5
Full model	92.8	91.5	93.1	90.0	25.4	15.3

According to Table 3, the accuracy of the complete model reached 92.8%, with an F1 score of 91.5%. The accuracy and recall were 93.1% and 90.0%, respectively, indicating the best performance. At the same time, the inference time was 25.4ms and the parameter count was 15.3M, which was slightly increased compared to other models in the ablation experiment. This was because the complete model combined multi-head AM, UBBiLSTM, and improved CNN, and each module brought higher complexity and computational complexity to feature extraction and representation. However, they were all within an acceptable range, indicating that the model achieved a balance between performance and efficiency. After removing the AM, CNN, or replacing MHA with single-head attention, the performance of the method decreased, indicating that each module played an important role in the method and jointly improved the classification performance.

On this basis, the study selected the Robustly Optimized BERT Pretraining Approach (RoBERTa), CNN for TC (TextCNN), and Hybrid Model Combining LSTM and CNN (LSTM-CNN) as comparative algorithms for short TC. The performance of these algorithms was first tested on the AG News Dataset. The result is shown in Figure 7.

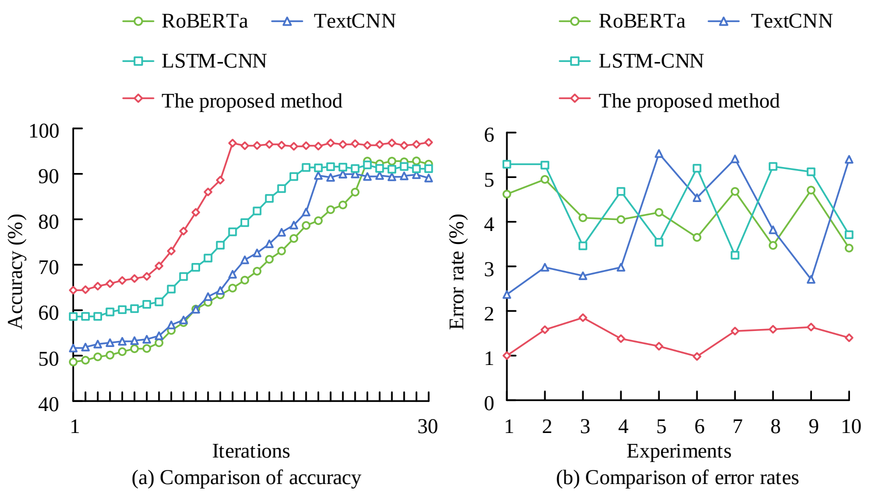


Figure 7: Comparison of short TC performance

As shown in Figure 7 (a), during training, the three comparison methods RoBERTa, TextCNN, and LSTM-CNN all achieved convergence after more than 20 iterations, with accuracy rates of 92.34%, 89.28%, and 91.62%, respectively. The method proposed in the study achieved convergence after 14 iterations, with an accuracy rate of over 96%. Meanwhile, the convergence speed was achieved with a batch size of 64, the use of cosine annealing learning rate scheduler, and the setting of early stopping criteria, indicating that this method had higher training efficiency under similar training configurations. As shown in Figure 7 (b), among the 10 experiments conducted on the validation set, the proposed method had the lowest classification error rate, averaging only 1.46%, while the error rates of the compared methods all exceeded 2%. From this, the proposed method exhibited higher efficiency and accuracy in short TC tasks. To quantify the performance advantage and reliability of this research method compared to the baseline model, the bootstrap method was used to calculate the 95% confidence interval of the accuracy of each model. The results are shown in Table 4. According to Table 4, the average accuracy of the proposed method was 89.2%, significantly higher than TextCNN's 78.3%, LSTM-CNN's 81.5%, and RoBERTa's 85.7%, and there was no overlap in the 95% confidence intervals of the four methods.

Table 4: 95% accuracy bootstrap confidence interva

Model	Average accuracy (%)	95% confidence interval (%)	Research method CI overlap
TextCNN	78.3	[76.8, 79.7]	No overlap
LSTM-CNN	81.5	[80.2, 82.7]	No overlap
RoBERTa	85.7	[84.9, 86.5]	No overlap
This research method	89.2	[88.8, 89.6]	/

At a 95% confidence level, the accuracy improvement of the proposed method was statistically significant compared to all baseline models, and the performance advantage was not caused by random fluctuations. Meanwhile, the confidence interval width of the proposed method was only 0.8%, indicating that its performance was less affected by data sampling differences and the results were more stable.

Furthermore, the classification performance of the testing method for long texts in IMDb was studied, and the Transformer model optimized specifically for long documents, Longformer, was added as a new comparative method. The results are shown in Figure 8.

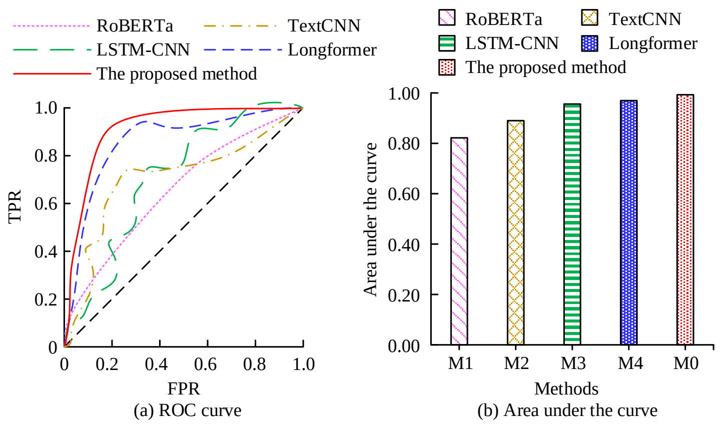


Figure 8: Long TC performance comparison

In Figure 8, a default probability threshold of 0.5 was used as the classification criterion. As shown in Figure 8 (a), the ROC curve of the proposed method almost completely covered the curves of other compared models, indicating that its classification performance was superior to other models at all thresholds. As shown in Figure 8 (b), the AUC value of the proposed method was close to 1.0, reaching 0.98. The AUC values of Longformer and RoBERTa were 0.95 and 0.92, respectively, indicating their strong performance in long text scenes, but still lagging behind the proposed method. The AUC values of TextCNN and LSTM-CNN were lower than 0.9. From this, the proposed method performed well in IMDb long TC tasks, and the BiLSTM method with AM could more effectively capture global semantic features in long texts. Furthermore, robustness testing was conducted against adversarial attacks. The widely used adversarial sample generation algorithm was used to generate adversarial samples, and the accuracy and robustness degradation rates on the adversarial samples were tested. The results are shown in Figure 9.

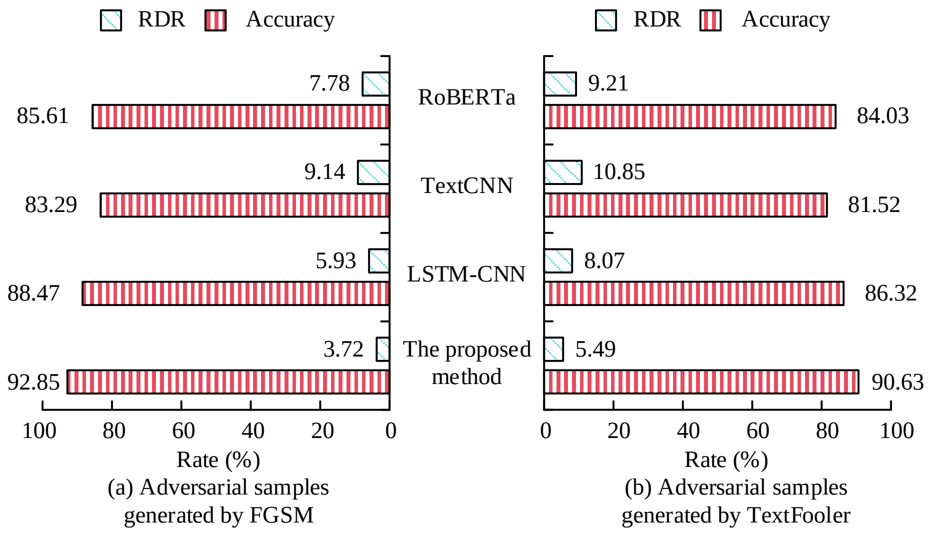


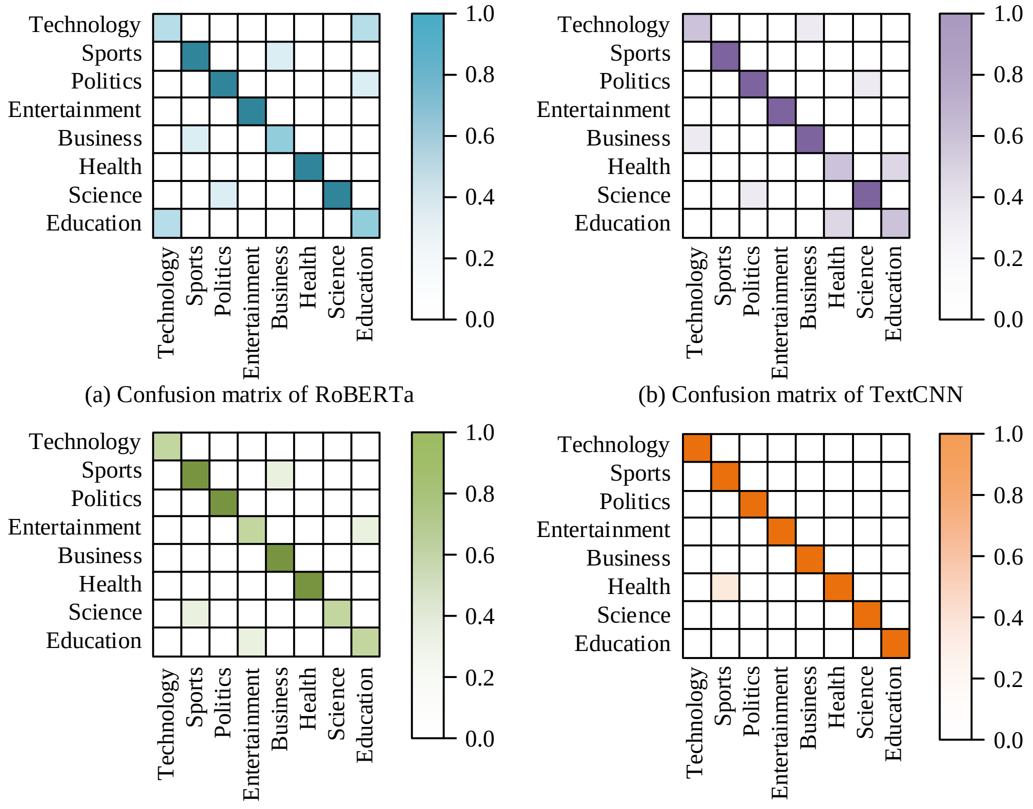
Figure 9: Robustness test results against attacks

Figures 9 (a) and 9 (b) show the test results in adversarial samples generated by the Fast Gradient Sign Method (FGSM) and TextFool, respectively. The FGSM attack disturbance budget ε used in the study was set to 0.01. In TextFool attacks, WordNet synonym filtering and semantic similarity ≥ 0.8 were used to ensure semantic preservation of adversarial samples. Both attacks were validated through manual sampling to avoid meaningless text perturbations. As shown in Figure 9 (a), the classification accuracy of the proposed method under FGSM attack was 92.85%, which was significantly superior to RoBERTa's 88.47%, LSTM-CNN's 85.61%, and TextCNN's 83.29%. In addition, the Robustness Drop Rate (RDR) was only 3.72%. According to Figure 9 (b), the accuracy of the proposed method under TextFool attack was 90.63%, higher than RoBERTa's 86.32%, LSTM-CNN's 84.03%, and TextCNN's 81.52%. The RDR was 5.49%, which was lower than the comparison method. From this, the proposed method, by integrating AM with UBBiLSTM, could more effectively capture key semantic information in text and reduce sensitivity to adversarial noise. To visually demonstrate the disturbance of text by adversarial attacks and the robustness of the proposed method to adversarial samples, two mainstream attack algorithms, Fast FGSM and TextFool, were selected and two qualitative examples were designed, as shown in Table 5.

Table 5: Qualitative examples of adversarial samples

Attack	Dimension	Content display
FGSM	Original text (Tag=1)	This movie's plot is clever, and the actor's performance is truly amazing—I would watch it again without hesitation.
	FGSM adversarial sample (after perturbation)	This movie's plot is clever, and the actor's performance is truly slightly amazing—I would watch it again without hesitation.
	Classification results of each model	TextCNN: Predict label=0 RoBERTa: predicted label=0 This research method: Prediction label=1
TextFooler	Original text (Tag=1)	The movie's pacing is too slow, the dialogue is boring, and I almost fell asleep halfway through—definitely not recommended.
	TextFooler adversarial sample (after perturbation)	The movie's pacing is too leisurely, the dialogue is dull, and I almost fell asleep halfway through—definitely not suggested.
	Classification results of each model	LSTM-CNN: Predict label=1 Longformer: Predict label=1 This research method: predicting label=0

According to Table 5, in FGSM attacks, the addition of "slightly" weakened positive semantics. TextCNN and RoBERTa mistakenly classified positive comments as negative, while the proposed method still captured the core positive statements and maintained the correct classification. In the TextFool attack, using synonyms to replace "slow" with "leisurely" and "boring" with "dull" weakened negative tendencies. LSTM-CNN and Longformer misjudged as positive, while the proposed method anchored negative semantics, resulting in accurate classification results. This indicated that the proposed method had stronger robustness to both gradient perturbations and semantic substitution perturbations. The next step in the research was to conduct cross domain generalization ability testing. The testing was conducted using zero sample transfer, where the model was trained on AG News and evaluated directly on the 20 Newsgroups dataset without any fine-tuning to test its original generalization ability. The model was trained using the AG News dataset and tested on a new news classification dataset of 20 Newsgroups (URL: http://qwone.com/ ~jason/20Newsgroups/), as shown in Figure 10.



(a) Confusion matrix of RoBERTa

(b) Confusion matrix of TextCNN

(c) Confusion matrix of LSTM-CNN

(d) Confusion matrix of the proposed method

Figure 10: Cross-domain generalization of proficiency test results

In Figure 10, the proposed method and all baseline models adopted a transfer approach of source domain fine-tuning combined with target domain testing. After completing model training and parameter optimization using the AG News dataset as training data, inference testing was directly conducted on the 20 Newsgroups dataset without using additional samples from the 20 Newsgroups dataset for fine-tuning. Grounded in the outcomes of the four confusion matrices in Figure 10 (a), (b), (c), and (d), the proposed method performed the best in cross domain generalization ability testing, with the most concentrated diagonal distribution of the confusion matrix and the least classification errors, demonstrating superior discriminative ability. In contrast, RoBERTa had more classification errors in Technology and Education, TextCNN performed poorly in Technology, Health, and Education, and LSTM-CNN, although better in some categories, had lower classification accuracy in Entertainment and Health. This indicated that the proposed method was more effective in capturing global and local features, and was suitable for TC tasks in different fields. To accurately quantify the classification performance of each model in cross domain scenarios, the overall accuracy, average F1 score for each category, and F1 score for key categories of each model on the 20 Newsgroups dataset were further calculated. The specific results are shown in Table 6.

Table 6: Quantitative results of cross-domain generalization test on 20 newsgroups dataset

Evaluation metrics	TextCNN (%)	LSTM-CNN (%)	RoBERTa (%)	The proposed method (%)
Overall accuracy (%)	78.3	81.5	85.7	89.2
Average F1 score for each category	76.5	79.8	84.2	87.6
Technology F1 score	72.1	75.3	80.5	85.8
Health F1 score	73.8	76.2	81.3	86.1
Entertainment F1 score	79.2	82.1	86.7	89.3
Education F1 score	79.4	81.8	85.9	88.7

According to Table 6, in the cross-domain testing of 20 Newsgroups, the overall accuracy of the proposed method reached 89.2%, which was 3.5% higher than RoBERTa, 7.7% higher than LSTM-CNN, and 10.9% higher than TextCNN. Meanwhile, the average F1 score of each class in this method reached 87.6%, with an F1 score of 89.3% in the Entertainment class and 88.7% in the Education class, significantly better than other models. The method proposed by the research could effectively capture the common semantic features of cross domain texts, reduce the impact of domain differences on classification performance, and improve generalization ability. To evaluate the actual deployment potential of the model, research was conducted on the Reddit-Multi-12K dataset (URL: https://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets ） to analyze the computational costs of each model. This dataset was mainly used for multi tag classification of forum posts. The computational efficiency comparison of different models on this dataset is shown in Table 7.

Table 7: Comparison of computational efficiency of different models

Model	Parameter quantity (M)	FLOPs (G)	Inference delay (ms/batch)	Memory usage (MB)
TextCNN	8.2	1.8	12.6	480
LSTM-CNN	12.5	3.2	19.8	650
RoBERTa	110.0	15.6	42.3	2800
This research method	15.3	4.5	25.4	820

According to Table 7, in terms of parameter count, TextCNN, as a lightweight model, had only 8.2M parameters, which was the smallest among all models. Due to the fusion of LSTM and CNN structures, the parameter count of LSTM-CNN was increased to 12.5M. However, RoBERTa, as a pre trained large model, had a parameter count of up to 110.0M. The parameter count of the research method was 15.3M, indicating that RoBERTa required much higher hardware resources for model storage and deployment than the research method. In terms of floating-point operations (FLOPs), TextCNN had an advantage with a low FLOP of 1.8G, while the FLOPs of the research method were 4.5G, but both were much lower than RoBERTa's 15.6G, indicating that the research method was more adaptable to low to medium computing hardware environments in terms of computational complexity. In terms of inference delay, the research method had a inference delay of 25.4m/batch, which was higher than TextCNN and LSTM-CNN, but could already meet the response requirements of most real-time TC scenarios. In terms of memory usage, the research method was 820MB, which was within the range of mainstream mid-range GPU's video memory capacity.

5 Discussion

5.1 Depth comparison with baseline model

The research method outperformed the baseline model in multiple tests. Compared to RoBERTa, the research method had fewer parameters, but performed better in robustness. The main reason was the reinforcement of core semantic features by UBBiLSTM and AM, rather than relying on rote memorization through large-scale pre training. Compared to Longformer, the research method had a higher AUC of 0.98 on IMDb long text tasks. The reason was that although Longformer's sparse attention was efficient, it might lose some global correlations in some cases, while the research method UBBiLSTM could more fully capture long-range dependencies.

4.2 Balancing model performance improvement and complexity

The ablation experiment showed that UBBiLSTM brought about an accuracy improvement of about 1.5% compared to standard BiLSTM, and the addition of AM also brought about a similar magnitude of improvement. Although the absolute improvement was limited on a high-performance baseline that was close to saturation, the true value of these improvements lied in robustness and generalization ability. In adversarial attack testing, the RDR of the proposed method was much lower than other models, proving that the semantic features it captured were more essential and stable. Therefore, the increased model complexity was worthwhile for building more reliable NLP systems in the real world.

4.3 Discussion on robustness and generalization mechanism

The high robustness and generalization ability of research methods mainly stemmed from their structural design. The improved CNN layer provided a robust local feature foundation. UBBiLSTM focused more on contextual information that contributed more to classification tasks through asymmetric mechanisms. The multi-head AM could still focus on the most core semantic signals in the text when facing adversarial noise or domain drift, thereby reducing interference and maintaining classification accuracy.

4.4 Limitations and future work

In addition to the above advantages, there are still some limitations in the research. Firstly, the dataset scope for experimental evaluation was limited, and future work should validate the model's generalization ability on a wider range of benchmarks. Secondly, although the computational cost of the model was lower than that of pre trained models, there is still room for optimization. Exploring lightweight methods such as knowledge distillation is an important direction for the future.

6 Conclusion

A TC method that integrated BiLSTM and AM was proposed to address the limitations of traditional TC methods in long texts and complex semantic scenarios. This method first utilized improved CNN and UBBiLSTM to extract global contextual information from text, and then focused on key features through AM to achieve effective feature fusion. In the ablation experiment, the complete model demonstrated superior performance, with an accuracy of 92.8% and an F1 score of 91.5%. All modules were pivotal. In the task of short TC, the proposed method performed best on the AG News dataset with an accuracy of 96% and a classification error rate of 1.46%, and its convergence speed was also faster than other models. In the task of long TC, the proposed method achieved an AUC value of 0.98 on the IMDb dataset, which was superior to advanced models such as Longformer and RoBERTa. In the robustness test of adversarial attacks, both FGSM and TextFool generated adversarial samples showed the lowest robustness degradation rate of the proposed method, with 3.72% and 5.49% respectively, demonstrating higher robustness. In the cross-domain generalization ability test, the proposed method showed the least classification errors on the 20 Newsgroups dataset, demonstrating superior cross domain adaptability. The results fully demonstrated that the proposed natural language TC method outperformed existing mainstream models in terms of performance, efficiency, and robustness. However, the proposed method had slightly higher complexity and inference time, which may pose application challenges in low computing scenarios. Future research will further explore lightweight model design and expand the application potential of methods in multilingual and multi-domain scenarios.

References

[1] Ullah A, Khan S N, Nawi N M. Review on sentiment analysis for text classification techniques from 2010 to 2021. Multimedia Tools and Applications, 2023, 82(6): 8137-8193. https://doi.org/10.1007/s11042-022-14112-3

[2] Parlak B, Uysal A K. A novel filter feature selection method for text classification: Extensive Feature Selector. Journal of Information Science, 2023, 49(1): 59-78. https://doi.org/10.1177/0165551521991037

[3] Wang Z, Li Q, Wang B, Wu T, Chang C Improving text classification through pre-attention mechanism-derived lexicons. Applied Intelligence, 2024, 54(22): 11765-11778. https://doi.org/10.1007/s10489-024-05742-1

[4] Reiss M V. Dissecting non-use of online news-systematic evidence from combining tracking and automated text classification. Digital Journalism, 2023, 11(2): 363-383. https://doi.org/10.1080/21670811.2022.2105243

[5] Li H, Yan Y, Wang S, Liu J. Cui Y. Text classification on heterogeneous information network via enhanced GCN and knowledge. Neural Computing and Applications, 2023, 35(20): 14911-14927. https://doi.org/10.1007/s00521-023-08494-0

[6] Jing T, Meng Q H, Hou H R. SmokeSeger: A Transformer-CNN coupled model for urban scene smoke segmentation. IEEE Transactions on Industrial Informatics, 2023, 20(2): 1385-1396. https://doi.org/10.1109/TII.2023.3271441

[7] Mohammed A, Kora R. An effective ensemble deep learning framework for text classification. Journal of King Saud University-Computer and Information Sciences, 2022, 34(10): 8825-8837. https://doi.org/10.1016/j.jksuci.2021.11.001

[8] Soni S, Chouhan S S, Rathore S S. TextConvoNet: A convolutional neural network-based architecture for text classification. Applied Intelligence, 2023, 53(11): 14249-14268. https://doi.org/10.1007/s10489-022-04221-9

[9] Jalal N, Mehmood A, Choi G S, Ashraf I. A novel improved random forest for text classification using feature ranking and optimal number of trees. Journal of King Saud University-Computer and Information Sciences, 2022, 34(6): 2733-2742. https://doi.org/10.1016/j.jksuci.2022.03.012

[10] Garrido-Merchan E C, Gozalo-Brizuela R, Gonzalez-Carvajal S. Comparing BERT against traditional machine learning models in text classification. Journal of Computational and Cognitive Engineering, 2023, 2(4): 352-356. https://doi.org/10.47852/bonviewJCCE3202838

[11] Yao J, Yuan B. Optimization strategies for deep learning models in natural language processing. Journal of Theory and Practice of Engineering Science, 2024, 4(5): 80-87. https://doi.org/10.53469/jtpes.2024.04(05).11

[12] Umer M, Imtiaz Z, Ahmad M, Nappi, M., Medaglia, C., Choi, G. S., & Mehmood, A. Impact of convolutional neural network and FastText embedding on text classification. Multimedia Tools and Applications, 2023, 82(4): 5569-5585. https://doi.org/10.1007/s11042-022-13459-x

[13] Kenarang A, Farahani M, Manthouri M. BiGRU attention capsule neural network for persian text classification. Journal of Ambient Intelligence and Humanized Computing, 2022, 13(8): 3923-3933. https://doi.org/10.1007/s12652-022-03742-y

[14] Enamoto L, Santos A R A S, Maia R, Weigang L, Filho G P R. Multi-label legal text classification with BiLSTM and attention. International Journal of Computer Applications in Technology, 2022, 68(4): 369-378.

https://doi.org/10.1504/IJCAT.2022.125186

[15] Ba S, Hu X, Stein D, Liu Q. Assessing cognitive presence in online inquiry-based discussion through text classification and epistemic network analysis. British Journal of Educational Technology, 2023, 54(1): 247-266. https://doi.org/10.1111/bjet.13285

[16] Khurana A, Verma O P. Optimal feature selection for imbalanced text classification. IEEE Transactions on Artificial Intelligence, 2022, 4(1): 135-147. https://doi.org/10.1109/TAI.2022.3144651

[17] Yan H, Gui L, He Y. Hierarchical interpretation of neural text classification. Computational Linguistics, 2022, 48(4): 987-1020. https://doi.org/10.1162/coli_a_00459

[18] Sobhanam H, Prakash J. Analysis of fine tuning the hyper parameters in RoBERTa model using genetic algorithm for text classification. International Journal of Information Technology, 2023, 15(7): 3669-3677. https://doi.org/10.1007/s41870-023-01395-4