applied sciences

MDPI

Article

Text Classification Model Based on Graph Attention Networks and Adversarial Training

Jing Li, Yumei Jian * and Yujie Xiong

School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China; 18773372067@163.com (J.L.); xiong@sues.edu.cn (Y.X.)
* Correspondence: jianyumei0628@126.com

Featured Application: Public Opinion Analysis.

Abstract: Text information on the internet often has a strong sense of immediacy, constantly reflecting societal dynamics and evolving events. This is especially crucial in the field of news text, where the classification and analysis of these immediate and varied text data become essential. Existing text classification models frequently struggle to effectively represent the semantic and local feature information of texts, limiting their effectiveness. The primary challenge lies in improving the representation of both semantic and local feature information in text classification models, which is critical for capturing the nuanced meanings in rapidly evolving news texts. This paper proposes a deep learning-driven framework designed to enhance the effectiveness of text classification models. The method incorporates noise perturbation during training for adversarial training, thereby enhancing the model's generalization ability on original samples and increasing its robustness. A graph attention network is employed to extract the contextual semantic information of vocabulary from sequential texts. This information is then combined with extracted sentence feature information to enrich the feature representation of the sequence. An attention mechanism is also introduced to extract more critical feature information from the text, thereby deepening the understanding of textual semantic information. Experimental results demonstrate that this method successfully integrates the boundary and semantic information of vocabulary into the classification task. The approach comprehensively and deeply mines the semantic features of the text, leading to improved classification performance.

Keywords: Chinese short text classification; graph attention networks; attention mechanism; adversarial training; feature fusion

Citation: Li, J.; Jian, Y.; Xiong, Y. Text Classification Model Based on Graph Attention Networks and Adversarial Training. *Appl. Sci.* **2024**, *14*, 4906. https://doi.org/10.3390/app14114906

Academic Editor: Valentino Santucci

Received: 22 May 2024
Revised: 31 May 2024
Accepted: 3 June 2024
Published: 5 June 2024

1. Introduction

With the rapid development of technology and the widespread adoption of the internet, people are increasingly active on social networking platforms. This shift has resulted in an abundance of news data [1], which often exhibit strong timeliness and contain potential economic benefits. For instance, e-commerce companies engage in opinion mining and sentiment analysis of user reviews to discern genuine customer needs and promptly refine their products. Such time-sensitive textual data can provide significant value to businesses, thereby making news text classification a popular area of research. Text classification approaches can be primarily divided into two major categories: traditional machine learning methods and deep learning-based methods. Compared to traditional machine learning algorithms, deep learning methods exhibit significantly enhanced expressive capabilities. Currently, the integration of deep learning with text classification has achieved substantial progress.

Traditional machine learning classification models primarily rely on feature engineering, including techniques such as the bag-of-words model [2] and n-grams. Representative algorithms in traditional machine learning include Naive Bayesian (NB) [3], Support Vector Machine (SVM) [4], and Maximum Entropy models [5].

Deep learning-based text classification models are capable of automatically extracting features and performing end-to-end learning, as well as capturing the underlying semantic information of texts. Mikolov et al. [6] introduced the Word2Vec model, which represents words as low-dimensional, dense vectors and uses the distances between these vectors to measure word similarity, thus capturing semantic information inherent to the words themselves. Kim et al. [7] proposed the TextCNN model for text classification, utilizing Word2Vec pre-trained word embeddings. This model has achieved favorable results in text classification tasks, although it falls short in representing local textual information and contextual relationships. Liu et al. [8] developed a Recurrent Neural Network (RNN) for text classification that is capable of capturing historical information in text sequences and effectively handling contextual data, thus aiding in the extraction of deeper textual features. However, RNNs tend to lose information when processing long texts. Yang et al. [9] proposed two types of attention-enhanced Bi-directional Long-Short Term Memory networks (Bi-LSTM) to improve classification performance, achieving notable results across multiple text classification tasks. Bahdanau et al. [10] introduced the attention mechanism into RNN models to address alignment challenges in sequence-to-sequence models and the poor performance with long texts. This mechanism [11] allows the model to handle varying input sizes and focus on the most relevant parts of the input, proving highly effective in tasks such as machine reading and learning sentence representations. The attention mechanism enhances the representation of weights in the feature learning process, clearly indicating the significance of each word in classification predictions, and has thus garnered extensive attention in the academic community. For example, Gu et al. [12] integrated the attention mechanism into a hierarchical multi-channel structure model to extract more significant subjective information from texts. Li et al. [13] proposed an end-to-end adversarial memory network that automatically captures keywords and further introduced the Hierarchical Attention Transfer Network (HATN), which enables the differentiation between core and non-core feature localization.

In recent years, Graph Neural Networks (GNNs) have also demonstrated commendable performance in text classification tasks. Kipf et al. [14] introduced the Graph Convolutional Network (GCN), which employs a semi-supervised approach to learn structural features within graphs and has achieved favorable results in node classification tasks. Yao et al. [15] developed the textGCN model, adapting the graph convolutional network for text classification by constructing texts as heterogeneous graphs, where nodes consist of words and documents, and edges are weighted by term frequency-inverse document frequency (TF-IDF) between documents and words, as well as pointwise mutual information between words, transforming the text classification task into a node classification challenge. Petar et al. [16] proposed the graph attention network (GAT), which utilizes an attention mechanism. This network employs masked self-attention layers to assign different weights to adjacent nodes, enhancing the model's ability to focus on more relevant features in the graph structure, thereby improving performance in tasks such as node classification within text-based applications. Liu et al. [17] propose a Deep Attention Diffusion Graph Neural Network (DADGNN) model that addresses limitations in existing GNN-based text classification methods, such as interaction difficulties between distant words and over-smoothing issues in deep graph layers. Lin et al. [18] introduces heterogeneous graph attention networks (HGATs) for semi-supervised short text classification. The model effectively incorporates different types of nodes and connections, which is particularly useful in scenarios with limited labeled data. Ai et al. [19] introduce a minimum-margin GAT designed to enrich feature information for short text classification, showing significant improvements over existing models like HGAT and STGCN. Li et al. [20] propose a graphic attention network text classification model that integrates label information, with a focus on integrating label semantic information into GATs for text classification, and enhancing the correlation between text and labels through attention mechanisms. Wang et al. [21] proposed using graph learning to fuse contextual information for text classification, which optimized the fusion of text graphs and contextual information to achieve better docu-

ment classification and demonstrated the effective integration of word interaction and graph learning.

Previous studies have acknowledged the significance of key vocabulary in sentence classification; however, existing models have not fully utilized the boundary information of characters and the semantic information of words. To further enhance the performance of text classification, this paper employs a graph attention network (GAT) that effectively integrates lexical knowledge. The contributions of this work are outlined as follows:

1. We utilize three distinct graph attention networks (GATs) to extract features from the contextual vocabulary of input text sequences. By concatenating these features with the original input text sequences, we achieve a superior representation of the text.
2. During the model training process, we introduce noise perturbations for adversarial training. Experimental results indicate that the incorporation of noise perturbations enhances the model's generalization ability on original samples and improves robustness.
3. We employ a multi-head attention mechanism, wherein the weight matrices assign higher numerical values to key information. The experiments demonstrate that this approach can further enhance classification accuracy.

The rest of this paper is organized as follows: Section 2 presents our network framework. Section 3 provides the experimental results, followed by an analysis of these results. Section 4 concludes the paper.

2. Materials and Methods

2.1. Model Framework

The framework of the text classification model based on graph attention networks and adversarial training is illustrated in Figure 1. The model consists of four components: the encoding layer, the graph attention layer, the fusion layer, and the attention layer. In the encoding layer, text is vectorized and perturbed with adversarial training to construct adversarial samples. TextCNN is used for feature extraction from the text, after which character embedding matrices and word embedding matrices are concatenated and fed into the graph attention network layer to capture contextual vocabulary information. The fusion layer integrates information from different network structures captured in the text sequences. An attention mechanism is introduced to assign varying weight values to the global sequence. Finally, the resultant feature representation is classified using a softmax function to determine the category with the highest probability label.

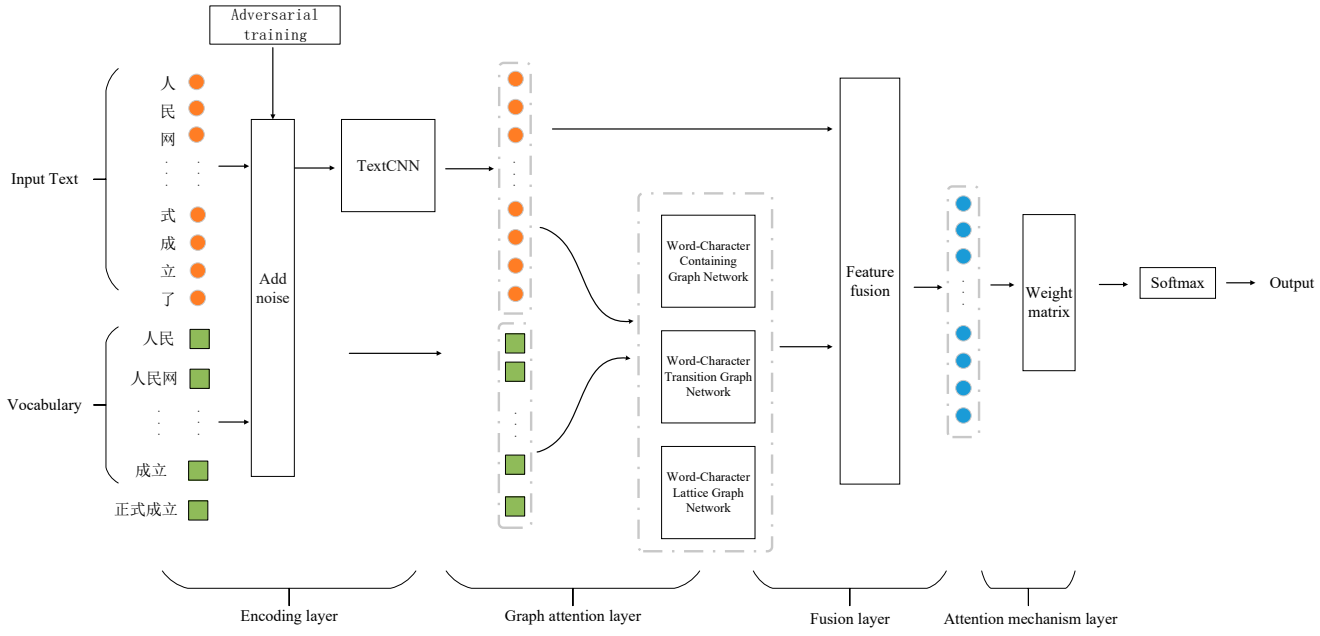

Figure 1. The overall architecture of our method.

2.1.1. Subsection Encoding Layer

In this paper, the model vectorizes text through a word embedding layer, generating input character vectors and input word vectors. During model training, an adversarial training method is incorporated, specifically the Fast Gradient Method (FGM) proposed by Goodfellow et al. [22]. The Fast Gradient Method (FGM) was selected for our adversarial training due to its efficiency and effectiveness in generating adversarial examples. FGM is computationally less intensive compared to other methods such as Projected Gradient Descent (PGD) or Carlini and Wagner (C&W) attacks, allowing for faster training times without significantly compromising robustness. Additionally, FGM's simplicity and strong performance in improving model generalization and resilience against adversarial attacks make it a preferred choice. By perturbing the input data with the gradient of the loss function, FGM effectively exposes the model to a variety of adversarial scenarios, enhancing its ability to generalize to unseen examples. The standard format of adversarial training is as follows:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\Delta x\in\Omega} L(x+\Delta x, y;\theta)\right], \tag{1}$$

In the formalism, $\mathcal{D}$ represents the training dataset, x denotes the input, y is the label, θ is the model parameters, and $L(x, y; \theta)$ is the loss for a single sample. Δx is the adversarial perturbation, and Ω is the space of perturbations. A common constraint is $\|\Delta x\| \leq \epsilon$, where ϵ is a constant. For each sample, an adversarial example $x + \Delta x$ is constructed, and the pair $(x + \Delta x, y)$ is used to minimize the loss and update the parameters θ via gradient descent. The adversarial perturbation Δx used is as follows:

$$\Delta x = \epsilon \nabla_x L(x, y; \theta), \tag{2}$$

For $\nabla_x L(x, y; \theta)$, standardize the following:

$$\Delta x = \epsilon \frac{\nabla_x L(x, y; \theta)}{\| \nabla_x L(x, y; \theta) \|}, \tag{3}$$

In our model, TextCNN is employed to extract features from the input text. TextCNN utilizes three convolutional layers with kernel sizes of three, five, and seven, respectively. Each convolutional layer considers a domain of the same size as its kernel, allowing the output to perceive a broader dimension of the input. The model's input is a sentence along with all the vocabulary of the contiguous sub-sequences matching the current sentence. We represent the input sentence as $s = \{c_1, c_2, \ldots, c_n\}$, where c_i denotes the i-th character, and the vocabulary matched to the sentence is represented as $l = \{l_1, l_2, \ldots, l_m\}$. Each character c_i is represented by a vector, denoted as $\boldsymbol{x_i}$, obtained by looking up the pre-trained character embedding matrix, where e^c is a lookup table for character embeddings.

$$\boldsymbol{x_i} = e^c(c_i) \tag{4}$$

By employing TextCNN on the sequence $\{x_1, x_2, \ldots, x_n\}$, we extract features, resulting in a sentence representation $H = \{h_1, h_2, \ldots, h_n\}$, where each h_i is a feature vector corresponding to the i-th position in the input sequence. To represent the semantic information of characters, we retrieve word embeddings from a pre-trained word embedding matrix. Each vocabulary item l_i is represented as a semantic vector, denoted by wv_i, where e^w is a lookup table for word embeddings. This representation facilitates a richer semantic understanding of the text, integrating both character-level and word-level information.

$$wv_i = e^w(l_i) \tag{5}$$

The sentence representation matrix H and the word embedding matrix wv_i are concatenated to form the output of the encoding layer. This combined matrix is denoted as $Node_f$, serving as a comprehensive feature set that captures both the contextual cues from

the sentence and the semantic attributes of the individual words. This concatenated matrix provides a robust foundation for subsequent layers to perform more sophisticated analyses and classifications.

$$Node_f = [h_1, h_2, \ldots, h_n, wv_1, wv_2, \ldots, wv_m] \quad (6)$$

2.1.2. Graph Attention Layer

To integrate information from self-matching vocabulary and the immediate contextual vocabulary, this paper employs three graph attention network layers to structure the model. These layers are as follows:

1. Word–Character Containing Graph Network: This network is designed to aid characters in acquiring boundary information and semantic information from self-matching vocabulary. It establishes connections between characters and their directly associated words, enriching the characters with deeper lexical insights.
2. Word–Character Transition Graph Network: This network captures the semantic information of the nearest contextual vocabulary. It transitions between characters and words that form contextual relationships, helping to understand the flow and connection of ideas within the text.
3. Word–Character Lattice Graph Network: Inspired by Zhang Yue's [23] use of a lattice structure in LSTM models to integrate vocabulary knowledge, this paper extracts the lattice structure to form the third attention network layer. This structure allows for a more flexible and interconnected approach to handling complex character–word relationships, providing a mesh-like framework that captures broader lexical fields.

Each of these network layers shares the same set of vertices, which consist of characters from the sentence and their matching vocabulary. However, the sets of edges are entirely distinct, facilitating specialized processing by each layer.

To represent the sets of edges, adjacency matrices are introduced. An element in an adjacency matrix indicates whether the vertices in the graph are adjacent; '1' denotes adjacency, while a '0' denotes non-adjacency. This matrix-based approach allows for efficient representation and processing of graph data, facilitating effective attention-based learning across the different layers.

In the Word–Character Containing Graph Network (WCCGN), characters within a sentence can capture the boundary and semantic information of self-matching vocabulary. As demonstrated in Figure 2, if a vocabulary item m contains a character n, then the corresponding entry in the adjacency matrix C for the WCCGN (m, n) is assigned a value of 1. This indicates a direct relationship where the vocabulary item encompasses the character, thus linking characters to the words they contribute to forming. This connection facilitates the effective capture of both boundary and deeper semantic layers of information, enhancing the text's representational richness in the network.

The Word–Character Transition Graph Network (WCTGN) facilitates the capture of semantic information from the nearest contextual vocabulary for each character. As illustrated in Figure 3, if a vocabulary item m or a character n matches the immediately preceding or succeeding subsequence of a character j, the corresponding entry in the adjacency matrix T for the WCTGN (m, j) or (n, j) is assigned a value of 1. This establishes a direct semantic link between characters and their adjacent vocabulary items, reflecting immediate linguistic contexts.

Furthermore, if a vocabulary item m is contextually related to another vocabulary item k as part of the preceding or succeeding context, the adjacency matrix T entry (m, k) is also assigned a value of 1. This linkage captures broader contextual relationships, ensuring that the semantic flow between closely related vocabulary items within the text is maintained and effectively represented in the model. This structure enhances the model's ability to comprehend and integrate contextual nuances, significantly improving its text classification capabilities.

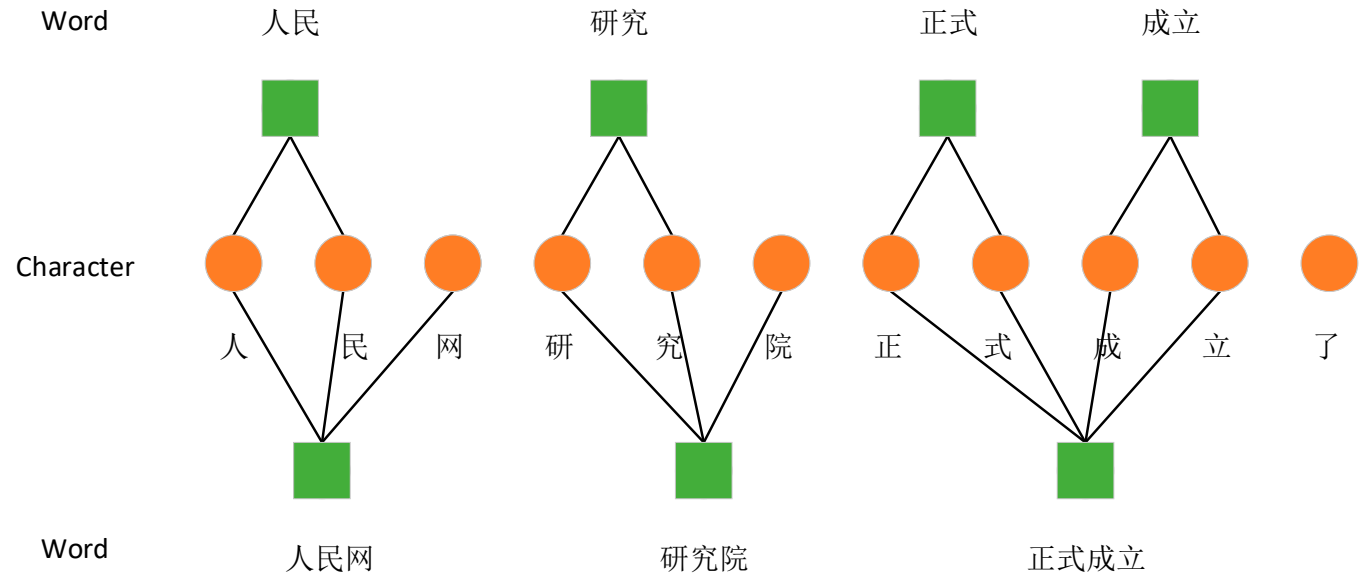

Figure 2. Word–Character Containing Graph Network.

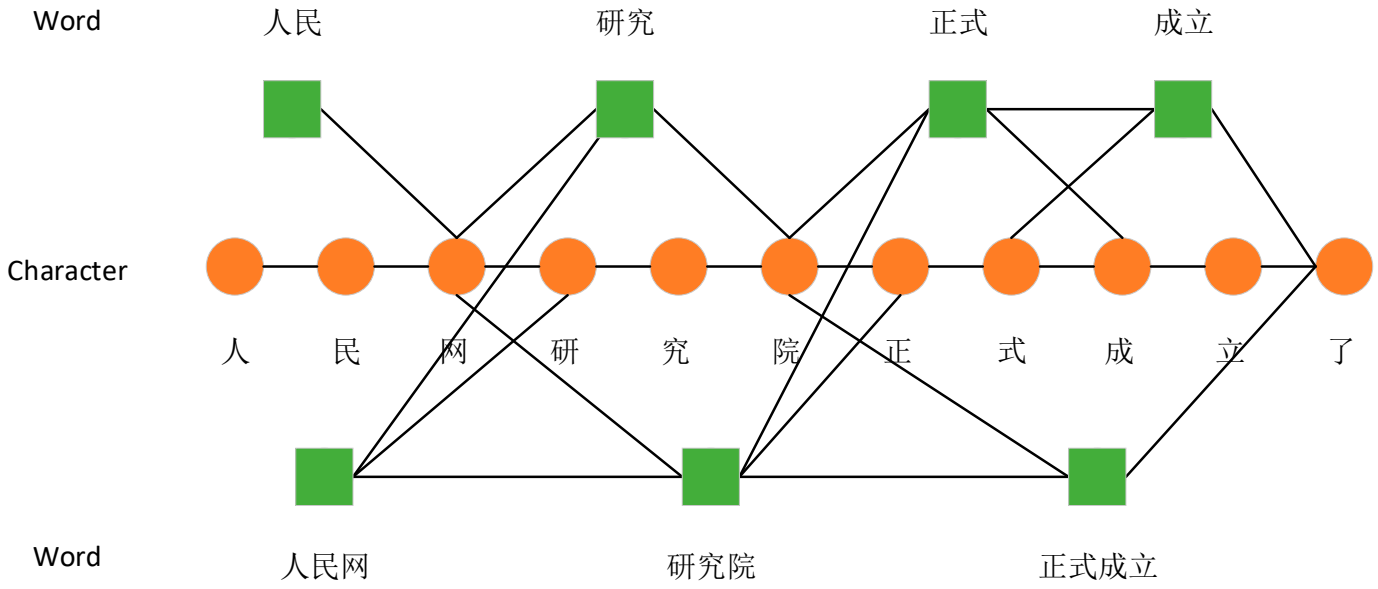

Figure 3. Word–Character Transition Graph Network.

The Word–Character Lattice Graph Network (WCLGN) is designed to implicitly capture the semantic information of the nearest contextual vocabulary and some self-matching vocabulary terms. As shown in Figure 4, if a character n is immediately preceding or succeeding another character j, the corresponding entry in the adjacency matrix L for the WCLGN (n, j) is assigned a value of 1. This assignment explicitly connects characters that are adjacent within the text, facilitating the capture of local contextual information.

Additionally, if a character n matches the starting or ending character of a vocabulary item m, the adjacency matrix L entry (n, m) is also assigned a value of 1. This linkage captures not just the adjacency but also the boundary alignment between characters and vocabulary terms. By mapping these relationships, the WCLGN can effectively integrate and represent both the direct context and the boundary information of vocabulary items, enhancing the model's capability to understand and process textual data comprehensively. This dual capture mechanism ensures that the text classification model can leverage both local and broader semantic cues efficiently.

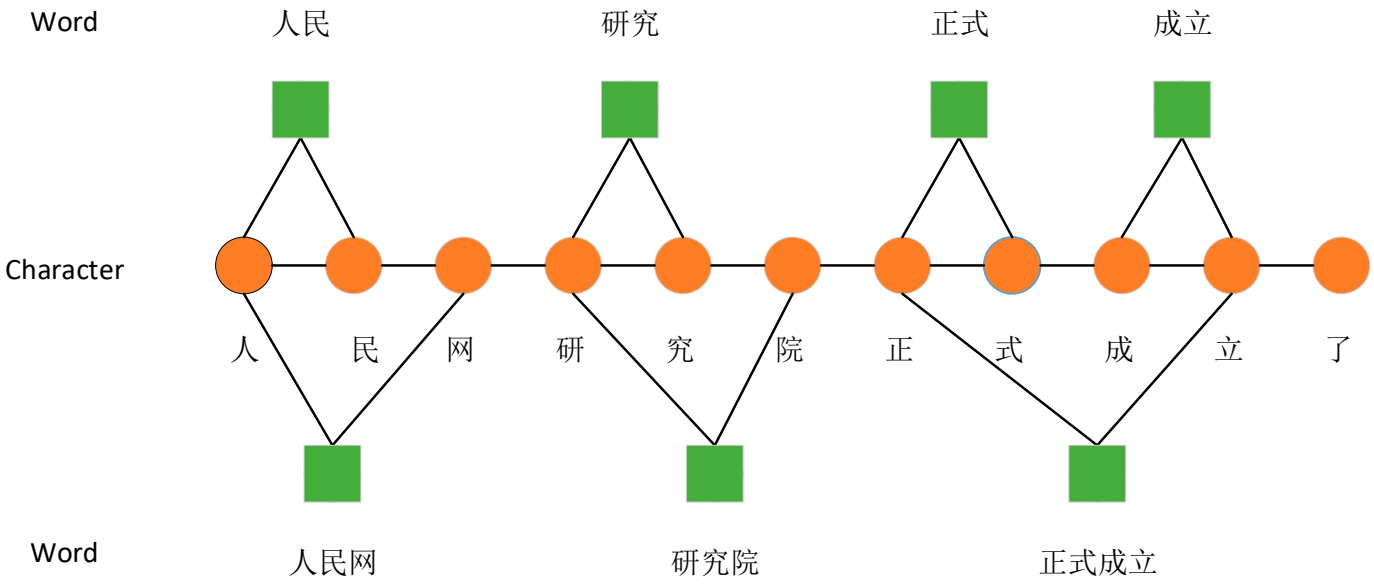

Figure 4. Word–Character Lattice Graph Network.

In the construction of the model using three graph attention network layers, each j-th layer of a graph attention network (GAT) receives an input set of node features $NF^j = \{f_1, f_2, \ldots, f_N\}$ along with an adjacency matrix $A \in \mathbb{R}^{N \times N}$, where N denotes the number of nodes and F represents the dimensionality of the features at the j-th layer. The output of the j-th layer is a new set of node features $NF^{(j+1)} = \{f'_1, f'_2, \ldots, f'_N\}$.

A GAT employs K independent attention mechanisms, each of which computes the importance of node i's features to node j's features. The attention mechanism in GAT is used to weigh the influence of each node's features on each other based on the structure specified by the adjacency matrix A. The computation of the new feature for each node involves aggregating features from its neighborhood, weighted by attention scores. The attention scores are computed as follows:

1. Linear Transformation: Each node feature f_j is first transformed by a weight matrix W, which is commonly shared across the network but specific to each attention head. This step projects the features into a space where attention coefficients can be more effectively learned:

$$f'_i = \|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k f_j\right), \tag{7}$$

2. Attention Coefficient Calculation: The attention coefficients that indicate the importance of node j's features for node i are calculated using a pairwise attention mechanism on the transformed features. Typically, this involves a nonlinear transformation such as the softmax function applied to a linear combination of features:

$$\alpha_{ij}^k = \frac{exp\left(LeakyReLU\left(a^T\left[W^k f_i \parallel W^K f_j\right]\right)\right)}{\Sigma_{k \in \mathcal{N}_i} exp\left(LeakyReLU\left(a^T\left[W^k f_i \parallel W^K f_k\right]\right)\right)}, \tag{8}$$

where a is a weight vector, $\|$ denotes concatenation, and $\mathcal{N}_i$ includes node i and its neighbors as specified by A.

3. Feature Update: The new features for each node are then computed as a weighted sum of the transformed features of the neighboring nodes, scaled by the computed attention coefficients:

$$f_i^{final} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k f_j\right), \tag{9}$$

where σ is an activation function.

This framework enables each node to dynamically adjust the influence of its neighboring nodes' features based on the overall structure of the graph, thereby effectively capturing both local and global structural information in the feature updates. This process is repeated for K attention heads, and the results can be averaged, depending on the specific architecture, to form the final output features for each node.

To construct models for three entirely distinct character–word graphs, this paper utilizes three independent graph attention networks (GATs), designated as GAT_1, GAT_2, and GAT_3. Since all three GATs share the same set of vertices, the input node features for each GAT are provided by the matrix $Node_f$. This shared matrix initializes the node features across all networks. The formula is as follows:

$$G_1 = GAT_1\left(Node_f, A^C\right), \tag{10}$$

$$G_2 = GAT_2\left(Node_f, A^T\right), \tag{11}$$

$$G_2 = GAT_2\left(Node_f, A^L\right), \tag{12}$$

Among them, $G_k \in \mathbb{R}^{F' \times (n+m)}$, $k \in \{1, 2, 3\}$, we keep the first n columns of these matrices and discard the last m columns because only these characters represent decoding labels.

$$Q_k = G_k[:0:n], k \in \{1, 2, 3\}, \tag{13}$$

2.1.3. Adjacency Matrix Processing

In the context of graph neural networks used for text classification, dealing with common issues like overfitting and oversmoothing is crucial, especially when handling complex data such as text. Overfitting occurs when a model performs excellently on training data but poorly on unseen test data, often due to insufficient training data or excessive model complexity, leading the model to learn the training data features too specifically at the expense of generalization. Oversmoothing, on the other hand, happens when an increase in network layers causes node representations to become overly similar, resulting in information loss and gradient vanishing.

To address these challenges, the Dropedge technique is introduced, which involves randomly removing some non-zero elements in the adjacency matrices and setting them to zero. This random alteration helps preserve original features while enhancing data diversity, which aids in reducing model complexity and preventing overfitting.

Furthermore, Dropedge can slow down the convergence of the network, which helps mitigate the oversmoothing issue. By reducing direct connections between nodes, the model can more effectively maintain crucial information from the input data and prevent homogenization of node representations, thereby enhancing the model's generalizability. Thus, Dropedge not only reduces computational complexity but also allows for deeper network layers, enabling the extraction of more profound features and enhancing the expressiveness of the network. This, in turn, improves the model's accuracy and robustness.

Figure 5 illustrates the process of applying Dropedge to the adjacency matrices, demonstrating how this technique modifies the network structure to improve model performance and stability in text classification tasks.

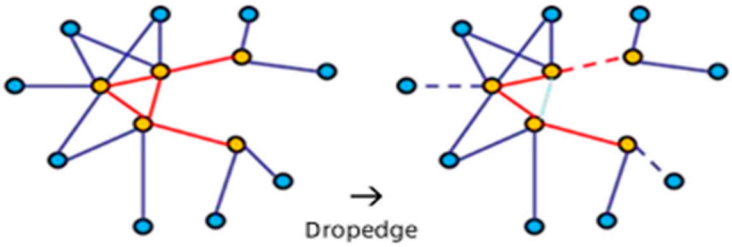

Figure 5. Perform Dropedge.

2.1.4. Fusion Layer

Local and global sequential information is captured through various network structures, integrating the sequential information extracted by TextCNN with the lexical information captured by three graph attention network layers. The input consists of the output H from TextCNN and the outputs Q_i from the three graph attention layers, i ∈ {1, 2, 3}. The formula is as follows:

$$R = W_1 H + W_2 Q_1 + W_3 Q_2 + W_4 Q_3, \tag{14}$$

In the fusion layer, W represents a trainable matrix. By applying this layer, we obtain a new matrix R, which serves as a new representation of the sentence. This representation encompasses semantic information from self-matching vocabulary and the nearest contextual vocabulary.

2.1.5. Attention Mechanism Layer

The essence of the attention mechanism is essentially a distribution of weight values, allocating larger weights to key parts. Even with longer texts, this mechanism enables the capture of essential elements, thus preserving important information and effectively enhancing classification performance. In this paper, the attention mechanism is employed to obtain the weight values of important vocabulary information within the input text. The formula is as follows:

$$Attention = tanh(inputs) * w, \tag{15}$$

where inputs represents a set of multiple vectors, $*$ represents matrix multiplication, w represents the parameter matrix to be trained, and the calculation formula for tanh is as follows:

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \tag{16}$$

The formula for determining the importance, $alpha$, of each position in the final vector is as follows: The term "mask" indicates whether padding has been applied at a particular position, with a value of 1 indicating padding and 0 indicating no padding. The purpose of applying the softmax function is to compute the importance weights of each position for the final vector, ensuring that the sum of these weights equals 1.

$$alpha = softmax(attention - (1 - mask) \times 10^8), \tag{17}$$

2.2. Experiments

2.2.1. Data Collection and Preprocessing

We conducted experiments to validate the effectiveness of the model on four benchmark datasets, THUCNews, Toutiao, Weibo, SougouCS, and Autohome. The preprocessing of all datasets is shown in Table 1.

Table 1. Datasets division.

Datasets	Class	Training	Test	Average Length
THUCNews	10	190,000	10,000	24.7
Toutiao	15	135,000	15,000	22.4
Weibo	2	89,736	10,000	57.1
SougouCS	12	28,347	6387	18.3

THUCNews: The THUCNews [24] news text dataset, provided and publicly released by Tsinghua University, is derived from historical data filtered from Sina News RSS subscriptions between 2005 and 2011. It comprises over 740,000 news documents across 14 news categories. For the purposes of this study, samples from 10 categories were selected, with 20,000 entries per category. Text lengths range from 20 to 50 words. Of these, 19,000 articles per category were used for the training set, and 1000 were used for the testing set.

Toutiao: The Toutiao [25] dataset is a Chinese short text classification dataset sourced from the Toutiao News app, containing a total of 382,688 news texts across 15 news categories. For this research, the data were randomly shuffled, and 150,000 texts were selected, with 135,000 used for the training set and 15,000 used for the testing set.

Weibo: This dataset was obtained from Sina weibo [26], where each sentence is marked as positive or negative. In this study, we randomly shuffled the data and selected 99,736 comment data. We used the Harbin Institute of Technology's word segmentation database for data preprocessing, with 89,736 comment data used in the training set and 10,000 comment data used in the testing set.

SougouCS: This dataset consists of Sohu [27] news titles from June to July 2012. We choose 12 types of news for experiments, including education, entertainment, technology, etc. This dataset consists of news statements from websites, which are relatively short and refined. The average length of the dataset is relatively short. This study extracted 34,734 statements, of which 28,347 were used for the training set and 6387 were used for the testing set.

This article uses the Harbin Institute of Technology's stop word list to process the dataset and obtain a vocabulary set. The specific information is shown in Table 1:

2.2.2. Hardware Configuration

In this paper, to verify the predictive performance of the model, both the model and benchmark experiments were conducted on the Windows 10 system using Python programming. The model was implemented using the PyTorch framework, and the main parameters of the model training computing environment were a GPU card (RTX 3070) and CUDA version 11.1. The experimental hardware configuration is outlined in Table 2 as follows:

Table 2. Experimental environment.

Parameter	Value
System	Windows10
Language	python
GPU	RTX 3070
CUDA	11.1

2.2.3. Parameter Settings

In this paper, Adam is employed as the optimizer for the model, which includes two layers of graph attention networks (GATs). Each GAT layer utilizes five attention heads and contains thirty units in the hidden layer. The dropout rate for the word embeddings is set at 0.4, and the maximum sequence length is limited to 128. The batch size is configured at 64, and the model undergoes training for a total of 50 epochs. The initial learning rate is set at 0.001. The parameters of the network model are outlined in Table 3 as follows:

Table 3. Parameter setting of model.

Parameter	Value
Batch_size	64
Learning_rate	0.001
Learning decay rate	0.01
Max_len	128
Drop_rate	0.4
Epoch	50
Optimizer	Adam

2.2.4. Comparison Model

This paper introduces several baseline sentiment analysis models, as follows:

In this paper, several text classification models are employed and assessed for their effectiveness:

1. TextCNN [7]: This utilizes Word2Vec to generate word vectors, followed by feature extraction through convolutional kernels of varying sizes. After passing through a max pooling layer, classification is performed using a Softmax function.
2. TextCNN+Att: After feature extraction using convolutional kernels of different sizes, the text passes through a weight adjustment layer that uses an attention mechanism to adjust weight values. This layer captures the dependency relationships between words based on the degree of influence of each word in the sequence on the text classification task and can learn the internal structural information of the sequence text.
3. TextRNN [8]: The use of RNN can effectively process sequence information and better extract contextual information features.
4. Transformer [28]: This comprises an encoder and decoder, using Word2Vec for converting input text into feature vectors. The self-attention mechanism effectively addresses the problem of long-distance dependencies in classification tasks.
5. BiLSTM [8]: This generates word vectors using Word2Vec and utilizes bidirectional LSTM layers to extract semantic information and dependencies in text, followed by classification through a fully connected layer.
6. HANs [29]: HANs is a Chinese short text classification model that uses CNN and BiLSTM to encode the character level and word-level features, respectively, and concatenates two-level features for classification.
7. CNN-highway + RNN [1]: This is a Chinese text classification model based on character-level features, where the sentence is encoded by highway-CNN and LSTM, respectively, and two outputs are concatenated for classification.
8. TextGCN [14]: Text-GCN constructs a heterogeneous graph based on the document and its words, and enables graph convolution networks to perform semi-supervised text classification.
9. Text-level-GNN [30]: Text-level-GNN is a text-level graph neural network model-based message passing mechanism, where a graph is constructed for each sentence and the words in the sentence are viewed as nodes.
10. RAFG [31]: RAFG is a Chinese text classification model that uses BiLSTMs to encode the Chinese radical information at character-level and word-level, respectively.
11. MACNN [32]: MACNN is a Chinese short text classification model to integrate character-level and word-level features by sentence-level attention mechanisms.
12. CW-GAT [33]: CW-GAT is a model that utilizes graph attention networks to effectively capture and integrate the interaction between character-level and word-level representations for improved Chinese text classification.
13. TextCNN+GAT+Att+FGM (TGAF): In this model, noise perturbation is added to the word embedding component for adversarial training. Adversarial samples are constructed during training to enable the model to correctly identify more adversarial examples, thereby enhancing the model's generalizability and robustness against adversarial attacks.

2.2.5. Evaluation Criteria

In this study, accuracy was used as an evaluation metric to assess the overall effectiveness of the classification. The terms are defined as follows: TP (True Positive) denotes correctly predicting a positive class as positive; FP (False Positive) denotes incorrectly predicting a negative class as positive; FN (False Negative) denotes incorrectly predicting a positive class as negative; and TN (True Negative) denotes correctly predicting a negative class as negative.

Accuracy is defined as the ratio of correctly classified texts ($TP + TN$) to the total number of texts ($TP + FP + FN + TN$), as shown in Equation (18):

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{18}$$

3. Results and Discussion

3.1. Sentiment Analysis Model Performance Evaluation

Firstly, the text sentiment analysis model proposed in this paper and the baseline models are evaluated from a quantitative analysis standpoint, using the evaluation metrics described in Section 2.2.5. All the models are trained for 40 epochs, and the best performances are recorded for a benchmark comparison. The specific experimental results of each model are shown in Table 4.

Table 4. Accuracy results of experimental algorithms.

	THUCNews	Toutiao	Weibo	SougouCS
TextCNN	0.9044	0.8530	0.9707	0.7976
TextCNN+Att	0.9129	0.8644	0.9815	0.8123
TextRNN	0.9102	0.8572	0.9736	0.8044
Transformer	0.9092	0.8661	0.9310	0.7553
BiLSTM	0.8515	0.7661	0.9767	0.7418
HANs	0.9159	0.8554	0.9647	0.8152
CNN-highway + RNN	0.8810	0.8053	0.9694	0.7117
TextGCN	0.9071	0.8713	0.8400	0.8105
Text-level-GNN	0.7995	0.8371	0.8354	0.7779
RAFG	0.8636	0.7805	0.9627	0.6795
MACNN	0.9214	0.8620	0.9650	0.8187
CW-GAT	0.9220	0.8738	0.9830	0.8262
TGAF	0.9301	0.8812	0.9874	0.8351

Clearly, the TGAF model performs better than all comparison models on several datasets. Specifically, on the larger THUCNews and Toutiao datasets, compared to the baseline model TextCNN, TGAF increased by 2.57 and 2.82 percentage points, respectively. TGAF also shows excellent performance on small-volume multi-classification datasets, such as the SougouCS dataset, which has improved by 3.75 percentage points compared to the baseline model TextCNN and 1.67 percentage points compared to the binary Weibo dataset, demonstrating the strong learning ability of the model.

When comparing Models 7 and 13, although CNN-highway + RNN utilizes both CNN and RNN for character feature extraction, it performs poorly in text experiments. This is because Chinese characters are polysemous, and the model fails to capture the word-level meanings within the characters, leading to inaccurate interpretations of the input sentences.

When comparing Models 6 and 11, which combine word-level features, we observed improved accuracy. This is because mixing hierarchical features enhances semantic information in Chinese, leading to better text comprehension. However, our model, through the use of graph attention networks (GATs), facilitates interaction between character and word information, resulting in a better understanding of the input text representation. By integrating the contextual semantics of the text and combining this information with the original input, our model deepens its analysis of the intrinsic meaning of the text.

When comparing Models 8 and 12, while CW-GAT and TextGCN utilize graph networks for feature extraction from input text and excel in capturing local features, they fail to fully leverage contextual information and integrate additional auxiliary features due to their structural characteristics. This limitation affects their performance in complex text classification tasks.

Comparing Model 1 and Model 2, TextCNN did not consider contextual features and did not introduce any auxiliary features, resulting in insufficient ability to capture features and poor classification performance. After adding an attention mechanism, the TextCNN+Att model showed a certain degree of improvement in performance. This is because the attention mechanism can enable important information in the text to receive higher classification decisions. Compared with Model 2 and Model 13, the introduction of a graph attention network in the model further improves the classification performance.

Three auxiliary features are introduced in the TGA model, making the model have better feature learning ability.

Additionally, our model introduces adversarial training to enhance performance and robustness. This method involves introducing finely-tuned perturbations to the training data samples to simulate potential external interferences, training the model to respond more stably to such disturbances. This not only improves the model's performance on the current dataset but also significantly enhances its generalization ability when handling novel or unseen data. The inclusion of adversarial training makes the model more robust, effectively identifying and resisting potential adversarial attacks or noise, ensuring consistent performance in dynamic environments. Moreover, this approach further boosts the model's generalization capability, allowing it to perform excellently on unseen data, demonstrating the crucial value of adversarial training in enhancing both robustness and generalization.

3.2. Ablation Experiment

To investigate the impact of the three distinct feature extraction capabilities of graph attention networks (GATs) incorporated into the model, ablation experiments were conducted using the THUCNews dataset and an automotive review corpus, as shown in Table 5. The ablation studies were performed by systematically removing different modules while keeping other components and parameters unchanged, to analyze their individual contributions to performance.

Table 5. Ablation experiment.

	THUCNews	Toutiao	Weibo	SougouCS
TGAF-GAT1	0.9250	0.8779	0.9844	0.8324
TGAF-GAT2	0.9223	0.8752	0.9826	0.8291
TGAF-GAT3	0.9253	0.8786	0.9853	0.8333
TGA	0.9288	0.8791	0.9862	0.8344
TGAF	0.9301	0.8812	0.9874	0.8351

In Table 5, TGAF represents the method proposed in this paper. TGAF-GAT1 indicates the removal of the Word–Character Containing Graph Network (GAT1); TGAF-GAT2 indicates the removal of the Word–Character Transition Graph Network (GAT2); TGAF-GAT3 indicates the removal of the Word–Character Lattice Graph Network (GAT3); and TGA represents the absence of adversarial training. The results show that removing any of the graph attention networks reduces the model's performance, underscoring the important role that each network plays in feature extraction.

Among these, the performance drop is most significant with TGAF-GAT2, which highlights the critical role of the Word–Character Transition Graph Network in capturing the contextual information of vocabulary. This suggests that GAT2 is particularly effective in uncovering hidden correlations between vocabularies, contributing significantly to the overall ability of the model to interpret and classify text based on nuanced semantic relationships. This insight underscores the value of integrating diverse graph-based approaches to enhance the depth and accuracy of text classification models.

Figure 6 shows the number of correctly classified news articles for each category by each model on the THUCNews dataset, displaying the test results of 1000 news texts for each category. The results show that the TGAF model performed better in classification, with over 950 correctly classified items in the education category. For categories similar to sports, the adversarial training added to the model exhibits strong robustness. The classification of other categories also demonstrated the effectiveness of the three graph attention networks added, and there were fewer instances of misclassifying different topic categories.

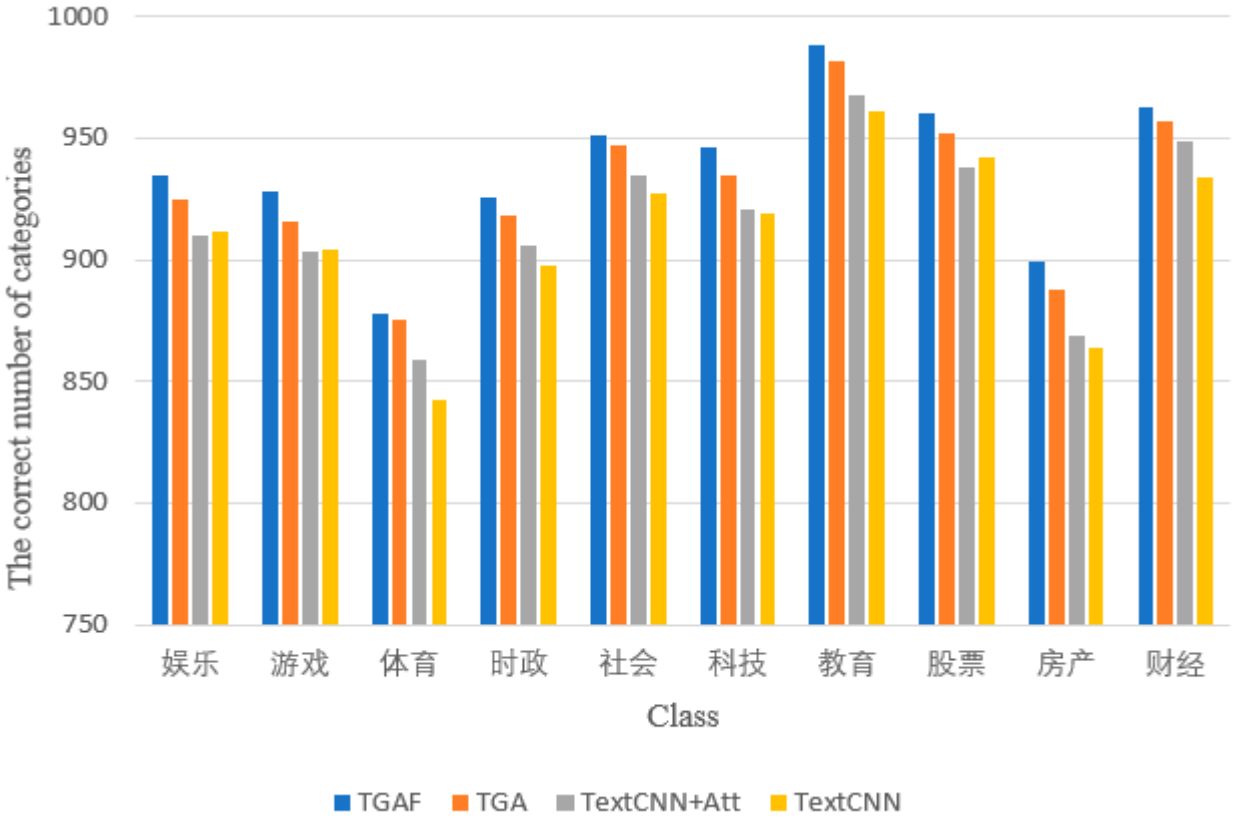

Figure 6. Number of correct classifications of model.

The size of the convolutional kernel is one of the key factors affecting model performance in text classification tasks. To further optimize model performance, this chapter conducts experiments to fine-tune the selection of convolutional kernel sizes. The results of the experiments are shown in Table 6. The model constructed in this chapter utilizes three CNN layers, necessitating a thorough investigation into the impact of various combinations of CNN kernel sizes on model performance. The first column in Table 5 lists different CNN size combinations; for example, (two, three, and four) indicates that the first convolutional layer of the model uses a kernel size of two, the second layer uses a size of three, and the third layer uses a size of four.

Table 6. Effect of convolution kernel size combination performance.

	THUCNews	Toutiao	Weibo	SougouCS
(2, 3, 4)	0.9271	0.8785	0.9831	0.8321
(3, 4, 5)	0.9277	0.8793	0.9844	0.8326
(2, 4, 6)	0.9284	0.8798	0.9852	0.8331
(3, 5, 7)	0.9301	0.8812	0.9874	0.8351
(3, 4, 6)	0.9288	0.8803	0.9856	0.8337

The experimental results demonstrate that different combinations of convolutional kernel sizes significantly affect model performance. For the text dataset, the optimal classification performance was achieved with a kernel size combination of (three, five, and seven). The size of the convolutional kernels plays a crucial role in the model's performance; smaller kernels may reduce the model's ability to extract features, thereby negatively impacting classification performance. Conversely, increasing the size of the convolutional kernels enhances the model's feature extraction capability but also increases the number of model parameters, adding to the model's complexity and potentially adversely affecting classification results.

In choosing the size of convolutional kernels, it is essential to balance the feature extraction capability and the complexity of the model to achieve the best text classification results. This process involves carefully considering the depth and breadth of feature detection required for specific types of text content while managing the computational load and the risk of overfitting.

4. Conclusions

This paper combines the advantages of graph attention networks (GATs) and adversarial training to propose a text classification model that employs TextCNN for extracting sequential text information and GATs for capturing the semantic information of textual context, leading to a richer integration of information. The model incorporates an attention mechanism to allocate varying weights to the global sequence representation. Experimental results demonstrate that the proposed text classification model achieves commendable results on both multi-class and binary classification datasets. With the inclusion of additional auxiliary features, the model can extract more features beneficial for classification, thereby enhancing its representational power. This study can help businesses understand consumer emotions and market trends, support decision-making, and optimize market strategies by categorizing social media comments, customer feedback, and product reviews.

There are still some limitations to the model in this article: the model is specifically developed for Chinese and may not be applicable to other languages, especially those with different syntactic structures and word orders, such as Japanese, Korean, Arabic, etc. The model needs to consider complex character combinations and contextual relationships when processing Chinese text, resulting in high computational complexity and resource consumption. There are many polysemous and ambiguous words in Chinese, which poses additional challenges to text classification. The model may require more contextual information to accurately classify.

One of the future research directions is to focus on multimodal analysis, which combines multiple data sources, such as images and videos, for comprehensive analysis. The combination of text, images, and videos can provide richer information sources and improve the accuracy of classification models. For example, in news classification, textual content can be analyzed together with relevant images and videos to obtain a more comprehensive understanding. Multimodal analysis faces challenges such as data alignment, feature extraction, and computational complexity. Future research needs to develop more efficient algorithms and models to address these challenges and fully leverage the advantages of multimodal data.

Author Contributions: Conceptualization, J.L.; Software, J.L.; Data curation, J.L.; Writing—original draft, J.L. and Y.J.; Writing—review and editing, J.L and Y.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chung, T.; Xu, B.; Liu, Y.; Ouyang, C.; Li, S.; Luo, L. Empirical study on character level neural network classifier for Chinese text. *Eng. Appl. Artif. Intell.* **2019**, *80*, 1–7. [CrossRef]
2. Harris, Z.S. Distributional Structure. *Word* **2015**, *10*, 2–162.
3. McCallum, A.; Nigam, K. A Comparison of Event Models for Naive Bayes Text Classification. In Proceedings of the AAAI-98 Workshop on Learning for Text Categorization, Madison, WI, USA, 26–27 July 1998; Volume 752, pp. 41–48.

4. Joachims, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In Proceedings of the 10th European Conference on Machine Learning, Chemnitz, Germany, 21–23 April 1998; Springer: Berlin/Heidelberg, Germany, 1998; pp. 137–142.
5. Xie, X.; Ge, S.; Hu, F.; Xie, M.; Jiang, N. An Improved Algorithm for Sentiment Analysis Based on Maximum Entropy. *Soft Comput.* **2019**, *23*, 599–611. [CrossRef]
6. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.S.; Dean, J. Distributed Representations of Words and Phrases and their Compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*.
7. Kim, Y. Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1746–1751.
8. Liu, P.; Qiu, X.; Huang, X. Recurrent Neural Network for Text Classification with Multi-Task Learning. *arXiv* **2016**, arXiv:1605.05101.
9. Yang, M.; Tu, W.; Wang, J.; Xu, F.; Chen, X. Attention Based LSTM for Target Dependent Sentiment Classification. In Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; AAAI: Menlo Park, CA, USA, 2017; pp. 5013–5014.
10. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
11. Raffel, C.; Ellis, D.P.W. Feed-Forward Networks with Attention can Solve Some Long-Term Memory Problems. *arXiv* **2015**, arXiv:1512.08756.
12. Gu, Y.; Yang, K.; Fu, S.; Chen, S.; Li, X.; Marsic, I. Multimodal Affective Analysis Using Hierarchical Attention Strategy with Word-Level Alignment. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018.
13. Li, Z.; Zhang, Y.; Wei, Y.; Wu, Y.; Yang, Q. End-to-End Adversarial Memory Network for Cross-Domain Sentiment Classification. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; pp. 2023–2030.
14. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2016**, arXiv:1609.02907.
15. Yao, L.; Mao, C.S.; Luo, Y. Graph Convolutional Networks for Text Classification. In Proceedings of the Thirty-First Innovative Applications of Artificial Intelligence Conference, Pasadena, CA, USA, 14–16 July 2009; AAAI Press: Menlo Park, CA, USA, 2019; pp. 7370–7377.
16. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
17. Liu, Y.; Guan, R.; Giunchiglia, F.; Liang, Y.; Feng, X. Deep Attention Diffusion Graph Neural Networks for Text Classification. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 7–11 November 2021.
18. Linmei, H.; Yang, T.; Shi, C.; Ji, H.; Li, X. Heterogeneous Graph Attention Networks for Semi-supervised Short Text Classification. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019.
19. Ai, W.; Wei, Y.; Shao, H.; Shou, Y.; Meng, T.; Li, K. Edge-enhanced minimum-margin graph attention network for short text classification. *Expert Syst. Appl.* **2024**, *251*, 124069. [CrossRef]
20. Li, J.; Qiu, M.; Zhang, Y.; Xiong, N.; Li, Z. A Fast Obstacle Detection Method by Fusion of Double-Layer Region Growing Algorithm and Grid-SECOND Detector. *IEEE Access* **2021**, *9*, 32053–32063. [CrossRef]
21. Wang, Y.; Wang, C.; Zhan, J.; Ma, W.; Jiang, Y. Text FCG: Fusing Contextual Information via Graph Learning for text classification. *Expert Syst. Appl.* **2023**, *219*, 119658. [CrossRef]
22. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards Deep Learning Models Resistant to Adversarial Attacks. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
23. Zhang, Y.; Yang, J. Chinese NER Using Lattice LSTM. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; Volume 1, pp. 1554–1564.
24. Dataset: THUCNews. Available online: http://thuctc.thunlp.org/ (accessed on 15 September 2022).
25. Dataset: Tutiao [DS/OL]. Available online: https://github.com/aceimnorstuvwxz/toutiao-dataset (accessed on 6 September 2022).
26. Dataset: Weibo2018. Available online: https://github.com/dengxiuqi/weibo2018 (accessed on 26 September 2018).
27. Dataset: SougouCS. Available online: https://tianchi.aliyun.com/dataset/94521 (accessed on 8 September 2022).
28. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
29. Zhou, Y.; Xu, J.; Cao, J.; Xu, B.; Li, C.; Xu, B. Hybrid attention networks for chinese short text classification. *Comput. Y Sist.* **2017**, *21*, 759–769. [CrossRef]
30. Kingma, P.D.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
31. Tao, H.; Tong, S.; Zhao, H.; Xu, T.; Jin, B.; Liu, Q. A Radical-Aware Attention-Based Model for Chinese Text Classification. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 5125–5132.

32. Hao, M.; Xu, B.; Liang, J.; Zhang, B.; Yin, X. Chinese short text classification with mutual-attention convolu tional neural networks. *ACM Trans. Asian Low Resour. Lang. Inf. Process.* **2020**, *19*, 61:1–61:13. [CrossRef]
33. Yang, S.; Liu, Y. A Character-Word Graph Attention Networks for Chinese Text Classification. In Proceedings of the 2021 IEEE International Conference on Big Knowledge (ICBK), Auckland, New Zealand, 7–8 December 2021; pp. 462–469. [CrossRef]