Text Classification Method Based on PEGCN

Zelin Guo[1], Ruidong Zhang[1], and Hai Huan[1]

[1]Nanjing University of Information Science and Technology

April 21, 2023

Abstract

The purpose of text classification is to label the text with known labels. In recent years, the method based on graph neural network (GNN) has achieved good results. However, the existing methods based on GNN only regard the text as the set of co-occurring words, without considering the position information of each word in the statement. Meanwhile, this method mainly extracts node features, but neglects the use of edge features between nodes. To solve these problems, a new text classification method, graph convolutional network using positions and edges (PEGCN), is proposed. In the word embedding section, a positional encoding input representation is employed to enable the neural network to learn the relative positional information among words. Meanwhile, the dimension of the adjacency matrix is increased to extract the multi-dimensional edge features. Through experiments on multiple text classification datasets, the proposed method is shown to be superior to the traditional text classification method, and has achieved a maximum improvement of more than 4%.

Introduction

Text classification is a core task of natural language processing and has been used in many real-world applications, such as spam detection[1] and opinion mining[2]. Transduction learning[3] is a special text classification method that uses both labelled and unlabeled samples in the training process. A graph neural network (GNN) is an effective transduction learning method[4,5] and is widely used in text classification applications. This method constructs a graph to model the relationship between documents. Nodes in the figure represent text units such as text or documents, while edges are constructed based on semantic similarity between nodes. Therefore, a GNN can be used to learn and classify nodes in the figure. The advantages of this method for classification are as follows: (1) The representation of each node depends not only on itself but also on its neighbors, endowing the representation of nodes certain context information; (2) During training, the model spreads the influence of supervision labels in training and test cases through graphs and edges. Even data with no labels help to represent the learning process, yielding a higher performance.

However, the use of GNN for text classification has the following problems: (1) The method based on GNN does not regard the text as a sequence but as a set of co-occurring words. In the task of text classification, the word order relationship in the sentence plays a crucial role in the final classification result; (2) The traditional GNN does not make full use of edge features. Only 0 and 1 are used between nodes to ascertain whether there is a connection, namely, the connectivity feature; however, the edge features of the graph often have rich semantic information, such as the type of connection between nodes, connection strength, *etc* ., and should be represented as continuous vector features rather than binary variables.

Recent studies have shown that large-scale pre-training models are effective for various natural language processing tasks, especially text classification tasks[6,7]. The pre-training model takes the unsupervised corpus as the training object and can learn the rich text semantics implied in the language. However, the methods used for transducing text classification tasks prior to 2020[4,5,13-17] did not consider the use of pre-trained models. It was not until 2021 when Lin *et al* . proposed BERTGCN[26], which combines BERT and GCN and demonstrates the effectiveness of pre-trained models in transductive learning.

Major contributions

The use of a graph convolutional network using positions and edges (PEGCN) model is proposed to address the aforementioned problems. Firstly, aiming at the problem whereby the traditional GNN ignores the relative position relation between words, the position information encoding is added into the word embedding part, so that the network can learn the position information between words. Secondly, in view of the insufficient use of edge features in GNN, the adjacency matrix is proposed to raise and normalize to extract multi-dimensional continuous features of edges. Finally, combining the advantages of the large-scale pre-training model, it can be proved that using the large-scale pre-training model is beneficial to the transduction learning through experiments. The key contribution of the work can be divided into the following parts:

(1) In this paper, we propose the PEGCN model, which effectively addresses the issue of disregarding text positional information in graph neural networks by utilizing input representations with positional information in the word embedding section; (2) The new model can contain multidimensional positive edge features, which overcomes the limitation that the traditional GNN can only process one-dimensional edge features and makes full use of the features of nodes and edges in the graph;

Related work

Text classification

Text classification is an important task in natural language processing, which is often applied to sentiment analysis, news filtering, spam detection, and other scenarios[8]. Text classification uses features to represent raw text and provides them as inputs to downstream classifiers. The most commonly used representation is Word2vec[9], which uses low-dimensional dense word vectors to represent words, but it ignores the semantic relationship between words, so it faces problems such as data sparsity and polysemy. In recent years, deep neural networks such as convolutional neural network (CNN) and recurrent neural network (RNN)[10]have been applied to extract contextual information and semantic representation from text. The results show that the performance is better than the traditional method. Kim *et al* . achieved good results in sentence classification by using different filters to extract multi-granularity feature sentences[11]. Sinha utilized bidirectional long and short-term networks to convert words into context embedded representations[12], enabling the network to learn contextual information in statements.

GNN

A GNN is a connection model that captures the dependency between graph nodes by connecting the edges of nodes[13,14,15], which can be roughly divided into graph convolutional networks[16,17] and graph attention networks[18,19]. In 2019, Yao *et al* . proposed text graph convolutional network (TextGCN)[4], which applies the GNN to the text classification task for the first time. TextGCN first constructs a symmetric adjacency matrix based on the given diagram, then fuses the representation of each node with its neighbors through convolution operations, and finally sends the representation of the node to the softmax layer for classification. However, TextGCN assigns the same weight to each node, which is inconsistent with the actual contribution of each node to the final classification. To solve this problem, Petar *et al* . proposed graph attention network (GAT)[18], which uses masked self-attention methods to assign different weights to each node according to the characteristics of adjacent nodes. The problem of using graph convolutional network for text classification is solved. Only the text information is considered when constructing the graph for the pre-ordering work, but the heterogeneous information such as text labels is ignored. In 2020, Xin*et al* . established a GNN based on label fusion[20]. This method combines label information by adding "text-tag-text" paths while constructing graphs, through which supervisory information can be transmitted more directly between graphs. Chang *et al* . designed a local aggregation function[21], which is a shareable non-linear operation for aggregating local inputs with disordered arrangement and unequal dimensions over non-Euclidean domains. It can fit non-linear functions without activation functions and can be easily trained using standard back propagation. In 2021, Wang *et al* . proposed a new short text classification method based on GNN the better to utilize the interaction between nodes of the same type and capture the similarity between short texts[22]. This method first models the short text dataset as a hierarchical heterogeneous graph, then dynamically learns a short

document graph to make label propagation between similar short documents more effective.

With the emergence of large-scale pre-training models in recent years, Devlin *et al* . proposed the pre-training model BERT (Bidirectional Encoder Representations from Transformers) based on self-attention mechanism[6]. BERT enhances a new representation of the input data at each layer of the encoder, obtaining a text representation with contextual information using multiple attention operations on different parts. Liu *et al* . made improvements on this basis[7], cancelling the next sentence prediction task, using more diverse data for training and achieving better results. Some recent studies combined GCN and BERT. Jeong *et al* . proposed a citation graph model for paper recommendation tasks[23], which combines the output of GCN and BERT to make the interaction between local information and global information conducive to downstream prediction tasks. Lu *et al* . established a BERT model based on graph embedding[24], which connects word embedding with node representation, and makes local information and global information interact through BERT, to determine the final text representation.

Method

This paper constructs a heterogeneous graph containing word nodes and document nodes. The proposed PEGCN model architecture proposed is shown in Figure 1. The leftmost part shows the input to the model. The present work adds Token Embedding and Position Embedding as the word vector. Then the model feeds the vector data to the GCN layer and BERT layer respectively. Finally, the output of the two layers is interpolated and sent to the softmax layer for classification. In the GCN part of Figure 1, two layers of GCN layers are used as the graph network. The bottom half of Figure 1 shows the BERT layer of the pre-training model.

Figure 1. PEGCN network.

Position coding

Previous network models that utilized One-Hot vectors as GCN inputs were unable to consider the relative positional information between words. In contrast to such inputs, the PEGCN model uses the sum of Token Embedding and Position Embedding as input word embeddings, which are sourced from BERT distributed representations[6]. The input representation of BERT is the sum of Token Embedding, Segment Embedding, and Position Embedding. The introduction of Segment Embedding in BERT is primarily for the next sentence prediction task. As the specific classification tasks in this study all involve single sentences, the Segment Embedding used to distinguish between the preceding and following sentences is considered redundant for this task. Therefore, only the sum of Token Embedding and Position Embedding is used to represent the network input in this study. Specific details are illustrated in Figure 2. The Token Embedding layer converts each word into a fixed-size vector that contains the semantic meaning of the text. In this study, the length and dimension of word vectors are both based on the BERT paper. Each word is converted into a 768-dimensional vector representation. Assuming a sentence length of 128, the sentence is represented as a (128, 768) matrix after the Token Embedding layer.

Figure 2. Input representation of PEGCN.

The network will learn a vector representation on each Position of the Position Embedding. The vector representation is coded as the information of the sequence order. The network will judge the relative position relationship of words in the sentence through the offset of each vector. The Position Embeddings layer is essentially a table measuring (128, 768) with the first row (when seen as a vector) representing the first position of the first sequence, the second row representing the second position of the sequence, and so on. The data of each row in this table are randomly generated at the beginning and updated with the training of the network. In the specific training, the network will also consider the batch size of the model batch_size, therefore, the Token Embedding and Position Embedding are represented as the tensor of (batch_size,128,768). When they are added together, the final input representation can be obtained. The word vectors obtained in this manner are used as the input representation for PEGCN document nodes in this study. The node embedding X for the document is represented as a matrix of dimensions ((ndoc+nword)

× d), where ndoc represents the number of document nodes, nword represents the number of word nodes, and d represents the dimensionality of the node embedding.

Edge features

GNN defines a graph as a set of $G = (V, E)$,$V$and E representing nodes and edges respectively. Nodes are divided into word nodes and document nodes, and the edges between word nodes are defined as Point-wise Mutual Information (PMI)[4]. The edge between the word node and the document node is defined as Term Frequency-Inverse Document Frequency (TF-IDF)[25]. The weight of two nodes i and $j$the edges between them is defined as:

where $A_{i,j}$ is the adjacency matrix, the word frequency denotes the number of times a word appears in the document, and inverse document frequency is the logarithm of the total number of documents over the number of documents containing the word. PMI is a popular word association measure, which can collect co-occurrence information on all documents in the corpus with a sliding window of fixed size and calculate the weight between two word-nodes. The PMI values for words$i$ and words j are calculated as:

where, $\#W(i)$ is the number of sliding windows containing words i in the corpus, $\#W(i,j)$ is the number of windows containing words i and j, $\#W$ represents the total number of sliding windows in the corpus. To extract multidimensional edge features, the $N \times N$ dimensional adjacency matrix $A_{i,j}$ is raised to an $N \times N \times P$ tensor$\hat{E}_{\mathrm{ijp}}$, where P represents the P dimensional features of the edge. The specific process is shown in Figure 3.

Figure 3. Upgrading of tensor dimension$A_{i,j}$.

The tensor changes with the training of the network, and the extra dimension is the newly learned weight. The adjacency matrix, after dimensionalization, represents edge features with the value of continuous multidimensional, which can make full use of edge features compared with traditional GNN. After network training is completed,$\hat{E}_{\mathrm{ijp}}$is normalized as follows:

The $||$ operator joins operations. Herein, the initial node feature of the graph X is defined as the identity matrix, that is, each word or document is represented as a one-hot vector. After two layers of GCN, X is sent to Softmax for classification:

Where, $E = D^{-1/2} E_{\mathrm{ij}} D^{-1/2}$ denotes the normalized edge feature matrix. W_0 and W_1 are the weight parameters of training respectively.The output of the GCN layer is considered as the final representation of the document, which is then fed to the Softmax layer for classification.

Combining BERT and GCN

Herein, BERT is trained in another part of the network as an auxiliary classifier[6]. Combining BERT on GCN can make the network combine the advantages of large-scale pre-training model, resulting in more rapid convergence and better performance. In terms of specific implementation, an auxiliary classifier is constructed by embedding documents X directly into the Softmax layer:

Finally, linear interpolation is used to combine the representation of BERT and GCN with:

The reasons for better performance through interpolation are: $\mathrm{Z}_{\mathrm{BERT}}$ acts directly on the GCN input to ensure that the GCN input is tuned and optimized towards the goal. This helps the multi-layer GCN model to overcome inherent shortcomings, such as gradient disappearance or over-smoothing[27], thus resulting in an improved performance.

Experiment

Introduction to datasets

Five common public datasets are used for experiments (Table 1): 20NG is a corpus containing 20 categories, with a total of 11,314 documents in the training set and 7532 documents in the test set; R52 and R8 are two subsets of the Reuters dataset (R8 has eight categories and is split into 5485 training documents and

2189 test documents; R52 has 52 categories, divided into 6532 training documents and 2568 test documents); Ohsumed is a database from the medical sciences that contains 23 categories. Herein, 7400 documents are selected, among which 3357 documents are in the training set and 4043 documents are in the test set. MR is a dataset of movie reviews for binary sentiment classification, where each review contains only one sentence. There are 5331 positive and 5331 negative comments in the corpus.

Table 1. Dataset information

20NG	R8	R52	Ohsumed	MR
Classes	20	8	52	23	2
Dos	18,846	7674	9100	7400	10,662
Train dataset	11314	5458	6532	3357	7108
Test dataset	7532	2189	2568	4043	3554
Nodes	61,603	15,362	17,992	21,557	29,426
Edges	26,990,597	3,504,462	4,406,322	8,066,963	1,927,676

Where, Classes represent the number of categories, Dos represents the number of documents, Nodes represent the number of graph nodes, and Edges represent the number of edges.

Parameter settings

The maximum sentence length is set to 128 and the batch size to 64; SGD is used as an optimizer to optimize all trainable parameters. The number of hidden layer units in GCN used herein is 200, the learning rate is 0.001, and parameter λ is set to 0.3 according to the experiment. The BERT learning rate is set to 1×10^{-5} and the dropout value is set to 0.5. The experiments in this article were conducted on a high-performance computer with an Nvidia T1 graphics card and 32 GB of RAM, using the PyTorch 1.5.0 framework and Python 3.6.

Experimental analysis

As the λ -value is a hyperparameter in the proposed model, to determine the best value of λ , the PEGCN model is used to conduct experiments on the four datasets (R8, R52, Ohsumed, and MR) with different λ -values. The experimental results are summarized in Table 2.

Table 2. PEGCN network λ value experimental accuracy comparison. metric: accuracy (%)

λ	R8	R52	Ohsumed	MR
0.1	97.67	95.83	72.12	89.48
0.2	97.94	96.14	73.02	88.80
0.3	**98.22**	**96.65**	**73.26**	**89.59**
0.4	98.04	96.11	72.67	88.89
0.5	98.17	95.02	72.30	87.51
0.6 0.7 0.8 0.9	97.62 98.02 97.94 97.58	94.55 95.37 95.64 95.17	72.69 72.89 72.30 72.47	87.23 89.53 86.94 86.86

As can be seen from the data in the table, PEGCN has the highest classification accuracy in each dataset when λ is 0.3, that is, when the weight ratio of GCN to BERT is 0.3/0.7. The accuracy of the four datasets is 98.22%, 96.65%, 73.26%, and 89.59%, respectively. Compared with the case of a λ- value of 0.5, the accuracy is 0.04%,1.63%, 0.96%, and 2.08% higher, so λ is set to 0.3 herein because the large-scale pre-training model can significantly improve the classification effect, and assigning a larger weight is conducive to improving the accuracy. For specific datasets, GCN is also required for further feature extraction, so the best effect arises when the weight ratio is 0.3/0.7.

Based on TextGCN, the proposed method delivers an improvement. A position graph convolutional network (PGCN) is a GCN model that adds position information. A position and Bert graph convolutional network (PBGCN) is a network model that combines BERT based on the addition of position information. Meanwhile,

experiments on GAT were conducted to verify the effectiveness of the proposed method. A position and Bert graph attention network (PBGAT) is a GAT network with position information and a model trained together with BERT. Finally, the method of PEGCN is GCN model using position information and edge features. The improved experimental results are listed in Table 3.

Table 3. Module validity experiment accuracy comparison. metric: accuracy (%)

Model	20NG	R8	R52	Ohsumed	MR
TextGCN	86.30	97.07	93.56	68.36	76.74
PGCN	87.44	97.81	94.90	72.08	87.62
PBGCN	88.16	98.17	93.73	71.93	88.83
PBGAT	87.04	98.13	95.60	71.61	87.70
PEGCN	**89.66**	**98.22**	**96.65**	**73.26**	**89.59**

The difference between PGCN and TextGCN is that PGCN includes sequence position information while TextGCN does not. In the 20NG dataset, PGCN is increased by 1.11% (an improvement of 0.74% on the R8 dataset; 1.46% on the R52 dataset; Ohsumed and MR increased by 3.72% and 10.88%). At the same time, position information is added into GAT for experiment (*i.e* . the PBGAT model in Table 3). As can be seen from Table 3, the classification accuracy of PBGAT is higher than that of PGCN on the five datasets; adding position information into the network can significantly improve the classification accuracy, especially in the sentiment classification dataset MR. According to the analysis presented herein, because the task of emotion classification is closely related to word order, the effect on MR is significantly improved: for example, "I like this actor but I don't like this movie" without the position information, the model cannot tell where the actor is in relation to the movie. Once the positions are reversed, the emotion of the whole sentence is reversed. Therefore, the improvement of sentiment analysis dataset is the most obvious, which shows the necessity of adding position information.

To verify the effectiveness of edge features in improving network performance, a PBGCN and a PEGCN are compared here. The PEGCN model processes the adjacency matrix based on PBGCN and makes full use of the multi-dimensional features of the edge. The only difference between the two is the processing of the edge features. As seen from Table 3, the classification accuracy of PEGCN on five benchmark datasets has been improved, and the optimal classification effect has been achieved on all datasets. In particular, for the 20NG and MR datasets, the accuracy is improved by 2.62% and 1.89%, respectively. The analysis of this study: Compared with the edge features represented by the adjacency matrix in the past, the tensor after dimension enhancement has richer semantics. Discretized points can only indicate whether there is a connection between nodes, that is, connectivity features; while the edge matrix after dimension enhancement has P additional dimensional features, which can be used to represent more information between nodes, such as connection types, connection strength, etc., making full use of edge features helps the network to train better node representations, thereby improving the classification task. It can be seen from Table 3 that the full extraction of edge features has a certain effect on GCN to improve the accuracy of text classification.

To make a comparison with similar models, BERTGCN[26] is used as the baseline model: BERTGCN is the model combining BERT and GCN. The model proposed herein is compared with the model also combined with BERT and GCN (Table 4).

Table 4. Comparison of Classification Accuracy of Similar Models. metric: accuracy (%)

Model	20NG	R8	R52	Ohsumed	MR
TextGCN (Yao *et al.*[4], 2019)	86.30	97.07	93.56	68.36	76.74
BERT (Devlin *et al.*[6], 2018)	85.30	97.80	96.40	70.50	85.70
BERTGCN (Lin *et al.*[26], 2021)	89.30	98.10	96.60	72.80	86.00

Model	20NG	R8	R52	Ohsumed	MR
BERTGAT (Lin *et al.*[26], 2021)	87.40	97.80	96.50	71.20	86.50
PEGCN (present work)	**89.66**	**98.22**	**96.65**	**73.26**	**89.59**

As seen from Table 4, the performance of BERTGCN and PEGCN is much better than that of the single TextGCN or BERT models, with an improvement of 2 to 4% in each of the five datasets. Compared with BERTGCN, PEGCN is 0.36% better than BERTGCN on the 20NG dataset. The R8 and R52 datasets are improved by 0.12% and 0.05%; Ohsumed and MR datasets are improved by 0.96% and 3.59%, respectively. Both models, PEGCN and BERTGCN, combine GCN and pre-trained models, but PEGCN outperforms BERTGCN in terms of classification performance. This study analyzes that BERTGCN also uses BERT input representation, so this is not the main reason for the difference between the two. BERTGCN still uses the adjacency matrix for edge feature processing, while this study uses the processed edge matrix, which can extract richer edge features and improve the representation ability of node representation. At the same time, it also reduces the appearance of discrete points and improves storage efficiency. Therefore, PEGCN's final classification results are superior to BERTGCN. The improvements made in TextGCN presented here are conducive to the improvement of classification accuracy and the competitiveness of the proposed model among similar models. As seen from Table 4, the classification accuracy of a series of combined models of TextGCN and BERT is generally higher than that of a single model of BERT or TextGCN. This indicates that the combination of TextGCN and large-scale pre-training model can significantly improve the classification accuracy, and the combination of TextGCN and pre-training model has significant advantages.

The further to prove the reliability of PEGCN, the proposed method is also compared with other classical methods (Table 5). Among them, the CNN proposed by Kim *et al*[11] . in 2014, LSTM is the long and short-term memory network[10], and Bi-LSTM is the two-way long and short-term memory network[12]. PTE is a network model based on word embeddings proposed by Tang*et al* .[28] that learns word embeddings based on heterogeneous text networks with words, documents and labels as nodes, and then averages the word embeddings into document embeddings for text classification. FastText is a simple and efficient classification method proposed by Joulin[29], which imparts the mean value of words or N-grams as documents and passes them to a linear classifier for classification. LEAM is an attention model based on tag embedding proposed by Wang *et al* .[30], which imparts words and tags into the same space for text classification. SGC is a simplified graph network proposed by Wu *et al* .[31]; SSGC is a spectrogram network proposed by Zhu*et al* .[32], which uses Markov diffusion nuclei to derive GCN, combining the advantages of spatial and spectral methods.

Table 5. Comparison of classification accuracy of different models. metric: accuracy (%)

Model	20NG	R8	R52	Ohsumed	MR
CNN (Kim *et al.*[11], 2014)	82.15	95.71	87.59	58.44	77.75
LSTM (Miyamoto *et al.*[10], 2016)	75.43	96.09	90.48	51.10	77.33
Bi-LSTM (Sinha *et al.*[12], 2018)	73.15	96.31	90.54	49.27	77.68
PTE (Tang *et al.*[26], 2015)	76.74	96.69	90.71	53.58	70.23
FastText (Joulin *et al.*[27], 2017)	79.38	96.13	92.81	57.70	75.14
LEAM (Wang *et al.*[30], 2018)	81.91	93.31	91.84	58.58	76.95
TextGCN (Yao *et al.*[4], 2019)	86.30	97.07	93.56	68.36	76.74
SGC (Wu *et al.*[31], 2019)	88.50	97.20	94.00	68.50	75.90
BERT (Devlin *et al.*[6], 2018)	85.30	97.80	96.40	70.50	85.70
BERTGCN (Lin *et al.*[26], 2021)	89.30	98.10	96.60	72.80	86.00
SSGC (Zhu *et al.*[32], 2021)	88.60	97.40	94.50	68.50	76.70
PEGCN (present work)	**89.66**	**98.22**	**96.65**	**73.26**	**89.59**

As seen from Table 5, PEGCN has achieved the optimal precision effect on the five classified datasets; GNN-based methods such as TextGCN, SGC, and SSGC are generally superior to traditional methods based on CNN or RNN. On the MR dataset, PEGCN offers the most significant improvement in classification accuracy, mainly because this dataset is extremely sensitive to position information, which is consistent with the conclusion above. The classification performance of PEGCN is better than that of BERTGCN across all datasets, not only on a single dataset, which proves the reliability of the model. In the GNN classification method, PEGCN shows significant improvement on the 20NG and Ohsumed datasets; because the average sentence length of these two datasets is greater than that of the other datasets, and the graph network is composed of word-document statistics, this means that longer text may generate more document connections passing through intermediate word nodes, which facilitates message passing through the graph and better performance when combined with GCN. This may also explain why the GCN model performs better than the BERT model on 20NG. For datasets with short documents, such as MR, the ability of graph structure is limited, but after adding position information and edge information to the graph network, it can be found that the classification accuracy is significantly improved. This finding also proves the importance of position information and edge information to the classification task.

Conclusion

This paper proposes the PEGCN model, which fully utilizes the advantages of large-scale pre-trained models and graph convolutional networks for text classification. The model first uses input representations with positional information to enable the network to learn relative position information between text; then processes the adjacency matrix to extract edge features fully; meanwhile, uses the BERT model for auxiliary training; finally, combines the predictions of the two models using linear interpolation for classification. Through a series of experimental designs and comparative analyses, it is found that the proposed method in this study outperforms other methods on five commonly used public datasets, achieving the highest classification accuracy and demonstrating the effectiveness of the model. Also, through a series of effectiveness experiments, it is proved that adding positional information and extracting edge features in graph convolutional networks are effective for improving classification accuracy. In future research, this study will further explore the improvement space of this network.

Data availability

The data sets used in this paper are all public data sets. For details, please refer to the website:*https://github.com/ZeroRin/BertGCN/tree/main/data*Specific figures are available at the website.

References

1. A. H. Wang, "Don't follow me: Spam detection in Twitter," *2010 International Conference on Security and Cryptography (SECRYPT)* , 2010, pp. 1-10.
2. Rushlene Kaur Bakshi, Navneet Kaur, Ravneet Kaur, and Gurpreet Kaur. Opinion mining and sentiment analysis. *In 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* , 2016. pages 452-455.
3. Vladimir N. Vapnik. 1998. *Statistical Learning Theory* . Wiley-Interscience.
4. Liang Yao, Chengsheng Mao, and Yuan Luo. 2019.Graph convolutional networks for text classification. *In Proceedings of the AAAI Conference on Artificial Intelligence* , **volume 33,** pages 7370–7377.
5. Liu, X., You, X., Zhang, X., Wu, J., Lv, P.: Tensor graph convolutional networks for text classification. *In: Proceedings of the AAAI Conference on Artificial Intelligence* , **vol. 34,**pp. 8409 – 8416 (2020)
6. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.(2018)
7. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and V eselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692. (2019)
8. C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in Mining text data. *Springer* , **2012,** pp. 163 – 222.

9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. *In: 1st International Conference on Learning Representations,* ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings (2013), http://arxiv.org/abs/1301.3781
10. Y. Miyamoto, K. Cho. Gated word-character recurrent language model. in Proc. *EMNLLP* , Austin, Texas, **2016,** pp. 1992-1997.
11. Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.
12. K. Sinha, Y. Dong, J. C. K. Cheung, and D. Ruths, "A hierarchical neural attention-based text classifier," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing,**2018** , pp. 817 – 823.
13. Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* , 20(1):61 – 80.
14. Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *In Advances in neural information processing systems* , pages 1024 – 1034.
15. Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826.
16. Thomas N Kipf and Max Welling. 2016a. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
17. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019, May). Simplifying graph convolutional networks. *In International conference on machine learning* (pp. 6861-6871). PMLR.
18. Petar Veli Pebbles ckovi´c, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Y oshua Bengio.2017. Graph attention networks. arXiv preprint arXiv:1710.10903.
19. Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit Y an Y eung. 2018a. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *In 34th Conference on Uncertainty in Artificial Intelligence 2018* , UAI 2018.
20. Xint, Y., Xu, L., Guo, J., Li, J., Sheng, X., & Zhou, Y. (2021, January). Label incorporated graph neural networks for text classification. *In 2020 25th International Conference on Pattern Recognition (ICPR) (pp. 8892-8898). IEEE.*
21. Chang J, Wang L, Meng G, et al. Local-Aggregation Graph Networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019, PP (99):1-1.
22. Yaqing Wang, Song Wang, Quanming Yao, Dejing Dou:Hierarchical Heterogeneous Graph Representation Learning for Short Text Classification. EMNLP (1) 2021: 3091-3101.
23. Jeong, C., Jang, S., Park, E., & Choi, S. (2020). A context-aware citation recommendation model with BERT and graph convolutional networks. Scientometrics, 124 (3), 1907-1922.
24. Lu, Z., Du, P., & Nie, J. Y. (2020, April). VGCN-BERT: augmenting BERT with graph embedding for text classification. *In European Conference on Information Retrieval* (pp. 369-382). *Springer,*Cham..
25. Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. 2008. Interpreting TF-IDF term weights as making relevance decisions. ACM Trans. Inf. Syst. 26, 3, Article 13 (June 2008), 37 pages. https://doi.org/10.1145/1361684.1361686
26. Lin, Yuxiao & Meng, Yuxian & Sun, Xiaofei & Han, Qinghong & Kuang, Kun & Li, Jiwei & Wu, Fei. (2021). BertGCN: Transductive Text Classification by Combining geri weis-corbley and BERT. 1456-1462. 10.18653 / v1/**2021** . The findings - acl. 126.
27. Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018a. Deeper insights into graph convolutional networks for semi-supervised learning. *In Proceedings of the AAAI Conference on Artificial Intelligence,***volume 32** .
28. Tang, J., Qu, M., & Mei, Q. (2015, August). Pte: Predictive text embedding through large-scale heterogeneous text networks. *In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1165-1174).
29. Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of tricks for efficient text classification. *In EACL* , 427 – 431. Association for Computational Linguistics.
30. Wang, G.; Li, C.; Wang, W.; Zhang, Y.; Shen, D.; Zhang, X.; Henao, R.; and Carin, L. 2018. Joint

embedding of words and labels for text classification. *In ACL* , 2321 – 2331.
31. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019, May). Simplifying graph convolutional networks. *In International conference on machine learning (pp. 6861-6871). PMLR.*
32. Zhu H, Koniusz P. Simple Spectral Graph Convolution[C]//*International Conference on Learning Representation* .**2021** .

Competing interests (mandatory)

The authors declare no competing interests.

Author contributions

R.Z. set the experimental strategies. Z.G. draft the main manuscript text. H.H., Z.G. designed and applied the experiments. All authors reviewed the manuscript. H.H. handled the process and paper publication issues.