



# Novel GCN Model Using Dense Connection and Attention Mechanism for Text Classification

Yinbin Peng<sup>1</sup> · Wei Wu<sup>1</sup> · Jiansi Ren<sup>1,2</sup> · Xiang Yu<sup>1</sup>

Accepted: 17 March 2024  
© The Author(s) 2024

## Abstract

Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) based text classification algorithms currently in use can successfully extract local textual features but disregard global data. Due to its ability to understand complex text structures and maintain global information, Graph Neural Network (GNN) has demonstrated considerable promise in text classification. However, most of the GNN text classification models in use presently are typically shallow, unable to capture long-distance node information and reflect the various scale features of the text (such as words, phrases, etc.). All of which will negatively impact the performance of the final classification. A novel Graph Convolutional Neural Network (GCN) with dense connections and an attention mechanism for text classification is proposed to address these constraints. By increasing the depth of GCN, the densely connected graph convolutional network (DC-GCN) gathers information about distant nodes. The DC-GCN multiplexes the small-scale features of shallow layers and produces different scale features through dense connections. To combine features and determine their relative importance, an attention mechanism is finally added. Experiment results on four benchmark datasets demonstrate that our model's classification accuracy greatly outpaces that of the conventional deep learning text classification model. Our model performs exceptionally well when compared to other text categorization GCN algorithms.

**Keywords** Densely connect · Graph convolution · Attention · Text classification

## 1 Introduction

The fundamental task of natural language processing (NLP) is text classification, which assigns text to a certain category based on its content. As a result, text classification technology has been extensively applied in a variety of domains, including opinion mining [1], spam classification [2], news filtering [3] and others. Text preprocessing, text representation, and

---

✉ Jiansi Ren  
renjsv@cug.edu.cn

<sup>1</sup> School of Computer Science, China University of Geosciences, Wuhan 430078, China

<sup>2</sup> Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430078, China

text classification [4] are the three basic phases of text classification. The most crucial phase is text representation.

In traditional text classification methods, bag-of-words features (such as TF-IDF [5] and N-gram [6], etc.) are frequently utilized as text representation. Then, popular classifiers are applied, including Naive Bayes (NB) [7], Support Vector Machine (SVM) [8], k Nearest Neighbor (KNN) [9] and others. Deep learning technology has advanced quickly when it comes to computer vision, speech recognition and other applications. It has also been effectively used for text representation. Currently, deep learning based text classification models mainly include: Convolutional Neural Network (CNN) [10, 11], Recurrent Neural Network (RNN) (such as long short-term memory network (LSTM) [12], and gated recurrent neural network (GRU) [13]) etc. As transformer [14] has been paid more and more attention by researchers, transformer-based text classification methods [15, 16] have also appeared. Transformer's self-attention mechanism can process sequential data including text well.

Graph Neural Network (GNN) [17], including graph convolutional network(GCN), has recently drawn a lot of interest. As opposed to conventional deep text classification models (like CNN and RNN), GNN/GCN describes the text as a collection of edge-connected word networks rather than as a sequence. As a result, GNN/GCN makes it simpler to remember the text's overall information when compared to CNN and RNN. Yao et al. [18] proposed Text-GCN, which is the first time GCN is used for text classification. Text-GCN considers both documents and words as nodes in a graph, thus constructing a graph for the whole corpus. Solving the text classification task by classifying nodes that represent text, Text-GCN can achieve good performance on the task with a 2-layer structure, but its performance does not improve when the number of layers increases. Although existing GCN models have shown better text categorization results, they still have certain issues:

- Shallow GCN networks cannot capture features between long-range nodes, while purely stacking the GCN layers will cause shallow features (e.g., words, phrases) to be lost, which is detrimental to text classification.
- Most of the existing GCN models build a graph for the whole corpus, which leads to the overly large graph consuming a lot of memory resource. Also, new samples outside the existing corpus cannot be predicted.

In this paper, aiming at the above problems, we propose a densely connected graph convolutional network (DC-GCN) with attention to text classification. First, a deep graph convolutional network (GCN) [19] is built for feature extraction. As the GCN network deepens, the nodes tend to be smooth [20] and shallow features will disappear, which has a negative effect on the classification result. To eliminate the effect, the dense connection method in DenseNet [21] is used in our model. In addition, dense connections can also generate features of different scales by multiplexing shallow features, which makes our model more friendly to text features of different scales. Moreover, we use a new way to build the graph. Instead of building the entire corpus into a single graph, we create a unique graph for each text. We also change the original Text-GCN node classification task back to the document classification task. Different from node classification, document classification needs to aggregate all the node features before the final classification [22]. As a result, we create an attention module that can both collect node information and determine the relative relevance of various node information. In summary, the following are this paper's main contributions:

- A new way of composition is devised to build a separate graph for each sentence.
- Extend dense connection to GCN network and propose a new DC-GCN network framework.

- The network employs an attention mechanism to automatically determine the relative relevance of various word nodes.

The portions of this essay are divided as follows: A study on text classification methods is performed in the second section of the essay. The third section introduces the paradigm put forward in this study. The experimental validation of the model put forward in this study continues in the fourth section. The conclusion, which follows in the fifth section, will summarize this paper's findings.

## 2 Related Work

### 2.1 Text Classification

In traditional text classification methods, the three steps are usually followed. The first step is text preprocessing, including word segmentation, data cleaning, removing stop-words, etc. The next step is to extract artificial features from the document (or any other text unit). Some popular artificial features include bag-of-words features and extended features. In the third step, the classifier uses these features as input to make predictions. The most commonly used classification algorithms include NB, SVM, KNN, etc. Although these traditional text classification methods have achieved good classification results. They commonly have some serious problems. First, manual feature extraction is time-consuming and laborious. Second, artificial features extracted from one domain cannot be well migrated to a different domain [23].

### 2.2 Deep Learning Methods

The issues with conventional text classification techniques can be resolved by using deep learning in this area. Deep learning methods have high transferability and can automatically extract text features. Currently, CNN, RNN, and their combination make up the classic deep learning text categorization models.

With its great local feature extraction ability, CNN plays an irreplaceable role in text classification. In shallow CNN [10, 24, 25] text classification models, a single-layer multi-channel text classification model proposed by Kim et al. [10] achieved the best classification performance at the time. Inspired by computer vision to improve the effect by deepening the network depth, some researchers attempt to achieve better text classification performance by deepening the depth of CNN. Therefore, a number of models based on deep CNN [11, 26] are used for text classification. Johnson et al. [26] proposed a pyramid-shaped deep CNN (DPCNN) text classification model that has become the representative model of deep CNN. In addition, different from the traditional word level text classification models, Zhang et al. [27, 28] proposed a char level text classification model, which encodes text with CNN at the char level and achieves excellent classification performance.

RNN can determine the sentence's long-term dependencies by using its chain structure. As a result, more studies favor using RNN to extract textual features. For example, Tai et al. [29] proposed a tree-structured LSTM to learn richer semantic features. Zhang et al. [30] proposed a new type of LSTM, which combines LSTM with reinforcement learning, and has been verified on benchmark datasets. Xu et al. [31] borrowed ideas from DenseNet, and proposed a multi-channel RNN model for text processing.

To further enhance the performance of text classification, some research has combined the benefits of CNN and RNN. Lai et al. [32] used recursive RNN based on CNN for text encoding, which reduced noise when traditional window CNN was used to extract features. Zhou et al. [33] combined CNN and LSTM. They sent the local features that CNN had extracted into LSTM for re-encoding so that LSTM could acquire long-term relationships from high-order sequence data. Liu et al. [34] used LSTM to re-encode phrase features extracted by CNN in both forward and reverse directions, which further improved the classification performance.

Due to the high performance of transformer in processing sequential information, a number of researchers have used it for text classification. Shi et al. [15] proposed a POS-aware and layer ensemble transformer, which can utilize the parts-of-speech information explicitly of text and combine output from multiple encoder layers for classification. Liu et al. [16] proposed a model that combines transformer and GCN, which can extract context and sequential information.

### 2.3 Graph Convolutional Neural Network

Recently, with the success of GNN in action recognition tasks [35], some text classification researchers have also turned their attention to GNN. Different from traditional deep learning text classification models (CNN and RNN), GNN treats the text as a set of edge-related word nodes and then learns the features of the nodes in a specific way. In 2016, Defferrard et al. [36] first used GNN for text classification tasks and reached cutting-edge efficiency. Since then, a lot of research has applied GNN to text classification. Peng et al. [37] converted the text into a graph of words, which can learn features of word nodes through GCN. They proved the advantages of graph convolution in collecting non-consecutive and distant semantics through experiments. Yao et al. [18] constructed the entire corpus as a graph, and shifted the text classification task into a graph node classification task, using GCN as a node propagation approach. Based on Yao et al., Liu et al. [38] constructed graphs from sequence, syntax and semantics, which further enriched the semantic information in the text representation. However, it takes a lot of memory of the computer to form a graph of the full content. Therefore, Huang et al. [39] didn't construct the entire corpus as a graph, but constructed a separate graph for each text. They verified the effectiveness of this approach through experiments.

## 3 Methodology

The framework of our model is shown in Fig. 1. For an input text sequence  $S = [w_1, w_2, w_3, \dots, w_n]$ , graph building and word embedding ensure that the data can be input to the GCN network and form initial features. Then through a densely connected deep GCN for feature extraction. Text features of different scales are extracted. An attention mechanism is designed to select the important features in the graph. Finally, a fully connected layer and a softmax function are used as classifiers to obtain the final classification results.

### 3.1 Graph Building

Text-GCN constructs a graph for the whole corpus (including document nodes and word nodes to be classified). Therefore, the constructed graph is very large and the model is consumed by a large amount of memory. Moreover, it is not possible to classify new samples because

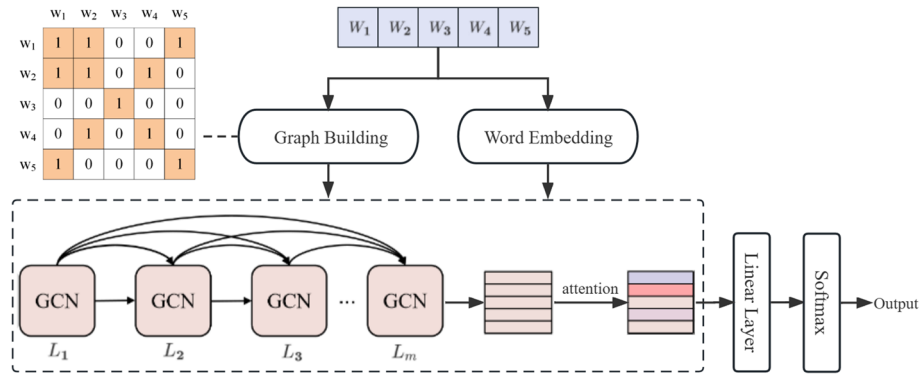


Fig. 1 Overall model framework

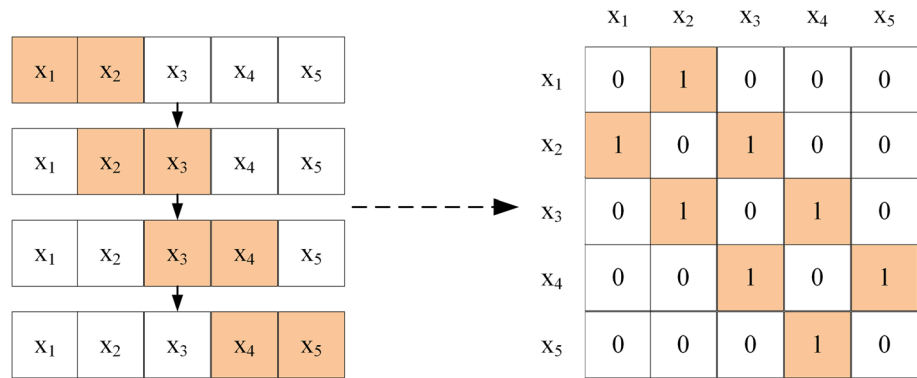


Fig. 2 The procedure for creating a graph. Assume that the sliding window's size  $p$  is 2. The adjacency matrix is shown in the figure on the right

the graph architecture and parameters depend on the existing corpus. To avoid the above shortcomings, we build a separate graph for each document and then train the model. In this way, a new graph can also be built for a new sample to complete its classification. A graph  $G = (V, E)$  for every text is built separately.  $V = [w_1, w_2, w_3, \dots, w_n]$  represents the set of all words in a text. The edge set  $E$  between vertices is determined by a sliding window  $p$ , similar to the sliding window in Text-GCN. Therefore, we can get the adjacency matrix  $A \in R^{n \times n}$  of graph  $G$ , where  $n$  is the number of word nodes. For example, given a sentence  $S = [w_1, w_2, w_3, \dots, w_n]$ , the sliding window  $p$  is 2. Then in the adjacency matrix, the corresponding value of  $w_1$  and  $w_2$  is 1, the corresponding value of  $w_2$  and  $w_3$  is 1, and so on. The specific process is shown in Fig. 2.

### 3.2 Word Embedding

Strings cannot be processed directly by a computer. In consequence of this, for a given string  $S = [w_1, w_2, w_3, \dots, w_n]$ , where  $n$  is the sentence length. We need to transform it into a format that computers can understand. Word embedding is a fundamental technique that can convert each word in the following statement into a fixed-dimensional word vector space:

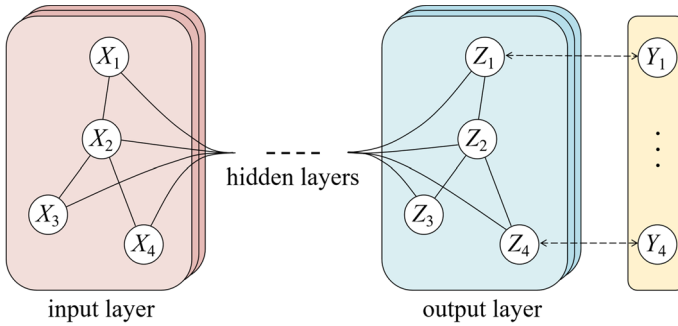


Fig. 3 Graph convolution structure

$$x_i = Ew_i \tag{1}$$

where  $w_i \in R^I$  represents the  $i$ -th word in sentence  $S$ .  $E \in R^{d \times I}$  is a word embedding dictionary matrix.  $I$  is the vocabulary size.  $d$  is the word embedding dimension. Therefore, the output of this layer  $X = [x_1, x_2, x_3, \dots, x_n] \in R^{n \times d}$  is a set of node vectors.

### 3.3 Densely Connected Graph Convolution

#### 3.3.1 Graph Convolution

The multi-layer GCN neural network is made up of the input layer, hidden layers, and the output layer. It makes use of word co-occurrence associations as edges, word co-occurrence vectors as nodes, and the adjacency matrix for learning node features. Figure 3 depicts the structure of a graph convolution.

Specifically, for the node feature matrix  $X = [x_1, x_2, x_3, \dots, x_n]$  and adjacency matrix  $A$ . Self-loops cause the diagonal elements of  $A$  to be set to 1. In addition, we also introduce a degree matrix  $D \in R^{n \times n}$ , where each element  $D_{ii}$  in  $D$  is:

$$D_{ii} = \sum_j A_{ij} \tag{2}$$

Therefore, for a GCN with one layer, the learning process of node features is:

$$L^{(1)} = \text{relu}(\tilde{A}XW_0) \tag{3}$$

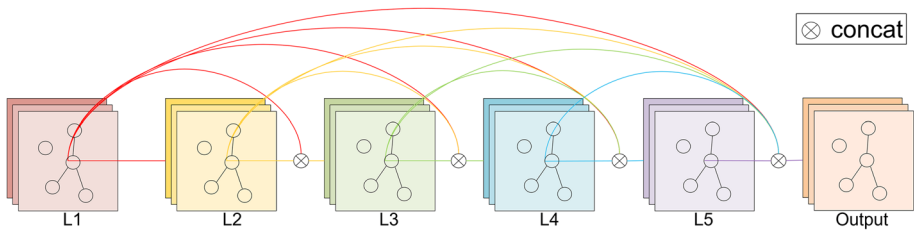
where  $\tilde{A} = D^{-1/2}AD^{-1/2}$ ,  $W_0$  is a trainable parameter.

#### 3.3.2 Dense Connection

It can be seen from Sect. 3.3.1 that a single-layer GCN can only capture the features of its neighboring nodes, Multi-layer GCN can capture the features of distant nodes through multiple neighbor nodes, which can get the key features in a multi-node graph. Multi-layer GCN stacking can be represented by the following formula:

$$L^{(m)} = \text{relu}(\tilde{A}L^{(m-1)}W_{m-1}) \tag{4}$$

where  $m$  is the number of layers of GCN.



**Fig. 4** Densely connect graph convolution structure.  $L_i$  is the  $i$ -th GCN layer, and  $L_1$  is the input through the first GCN layer

However, the shallow features can be lost as the network becomes deeper. Dense connections can directly combine shallow features with deep features. Inspired by DenseNet, we design a densely connected GCN network (DC-GCN), which can better capture node features at longer distances by multiplexing the features of shallow GCN layers. Figure 4 displays the structure of DC-GCN.

Therefore, the  $m$ -th layer propagation process of DC-GCN is:

$$L^{(m)} = \text{relu} \left( \tilde{A} \left[ L^{(1)} \oplus L^{(2)} \oplus \dots \oplus L^{(m-1)} \right] W_{m-1} \right) \tag{5}$$

where  $\oplus$  means splicing. Additionally, the prior GCN models only use the output of the final layer, which allows for the retention of only the scale properties of the final layer. Unlike the traditional method, we keep the results from each layer and used the intermediate features at various scales as the final output:

$$h_i = L_i^{(1)} \oplus L_i^{(2)} \oplus \dots \oplus L_i^{(m)} \tag{6}$$

$$H = [h_1, h_2, \dots, h_n] \tag{7}$$

where  $h_i$  is the  $i$ -th word node. Therefore, through this layer, we get the final feature matrix  $H \in R^{n \times (k \cdot m)}$ , where  $k$  is the dimension of GCN.

### 3.4 Attention Mechanism

The retrieved characteristics must be combined into a fixed-length vector to be classified. For the feature matrix  $H = [h_1, h_2, h_3, \dots, h_n]$ , not every feature contributes equally to the task. As a result, we create the attention module to highlight their significance. For each feature  $h_i$ , we calculate an attention score  $a_i$ ,  $a_i$  represents the importance of the feature to the classification task:

$$a_i = \frac{\exp(e_i^T u_s)}{\sum_j \exp(e_j^T u_s)} \tag{8}$$

$$e_i = \text{relu}(W_s h_i + b) \tag{9}$$

where  $W_s$  and  $u_s$  are trainable parameters, and  $b$  is the bias term.

Finally, we multiply the score on attention  $\alpha = [a_1, a_2, a_3, \dots, a_n] \in R^n$  and the feature matrix  $H$ . Sum and get the finished text representation  $V$ :

$$V = \sum_{i \in n} a_i h_i \tag{10}$$

Through this layer, we get the final text representation  $V \in R^{km}$ .

### 3.5 Classification

The classification module uses one fully connected layer. The purpose of this layer is to calculate the probability distribution of the category according to the final text representation  $V$ . This layer is represented by the following equation:

$$P = \text{softmax}(WV + b) \quad (11)$$

where  $P$  is the category's probability distribution.  $W$  is the trainable parameter, and  $b$  is the bias term.  $\text{softmax}$  is the activation function.

### 3.6 Implementation

The graph  $G = (V, E)$  is created for each text individually, where the  $V = [w_1, w_2, w_3, \dots, w_n]$  represents a text's entire word list. The sliding window  $p$  is 2, and the edge set  $E$  between vertices by the sliding window  $p$ . Then through the word embedding layer, the output of this layer  $X = [x_1; x_2; x_3; \dots; x_n] \in R^{n \times d}$  is a set of node vectors. In the DC-GCN block, 5 GCN layers are connected densely for feature extraction, and the connection method is concat. To select task-friendly features for text classification, we calculate an attention score  $a_i$  for each feature  $h_i$ ,  $a_i$  indicates the feature's significance to the categorization task. Then multiply the attention score  $\alpha = [a_1; a_2; a_3; \dots; a_n] \in R^n$  and feature matrix  $H$ . Sum and get the final text representation  $V$ .  $V$  is sent into the classifier to get the judged label, which has a fully connected layer and a softmax function. Algorithm 1 displays the general flow of the proposed procedure.

---

#### Algorithm 1 Proposed Model

---

**Require:** text sequence  $S = [w_1, w_2, w_3, \dots, w_n]$

**Ensure:** predictions for the test dataset's labels

- 1: Set batch size to 64, optimizer Adam (learning rate:  $1 \times 10^{-3}$ ), epochs number  $n$  to 100;
  - 2: Construct the graph  $G = (V, E)$  for every text separately;
  - 3: Create train loaders and test loaders.
  - 4: **for**  $i = 1$  to  $n$  **do**
  - 5: Obtain  $X = [x_1, x_2, x_3, \dots, x_n]$  by word embedding:  $x_i = Ew_i$ ;
  - 6: Obtain  $L^{(m)} = \text{relu}(\hat{A} [L^{(1)} \oplus L^{(2)} \oplus \dots \oplus L^{(m-1)}] W_{m-1})$  with 5 DC-GCN layers;
  - 7: Compute attention coefficient matrix:  $\alpha = [a_1; a_2; a_3; \dots; a_n] \in R^n$ ;
  - 8: Multiply attention score and feature matrix:  $V = \sum_{i \in n} a_i h_i$ ;
  - 9: Obtain classification results by linear layer:  $P = \text{softmax}(WV + b)$ ;
  - 10: **end for**
  - 11: Using the trained model with the test dataset, predict labels;
- 

## 4 Experiment

The experimental datasets, parameter settings, and comparative models are described in this section. We will contrast and evaluate the results of the experiment for the proposed model and the other models.



**Table 1** Dataset statistics

Dataset	Train	Test	Class	Avg.length
MR	7108	3554	2	20.39
R8	5485	2189	8	65.72
R52	6532	2568	52	69.82
Ohsumed	3357	4043	23	135.82

## 4.1 Dataset

This work chooses 4 benchmark datasets for experimental validation to assess the performance of the proposed model on the task of text classification. They include: MR dataset<sup>1</sup> [40], R8 and R52 dataset,<sup>2</sup> Ohsumed dataset.<sup>3</sup> Details of each dataset are as follows:

**MR:** A dataset of movie reviews with only one sentence per review. It contains 2 non-intersecting categories.

**R8:** A portion of the 21578 dataset from Reuters. It contains 8 non-intersecting categories.

**R52:** A portion of the 21578 dataset from Reuters like R8. But it contains 52 non-intersecting categories.

**Ohsumed:** A subset extracted from the MEDLINE dataset. It contains 23 non-intersecting categories.

Table 1 displays comprehensive statistics for each dataset. We chose 10% of each training set as the validation set at random.

## 4.2 Experimental Setup

### 4.2.1 Experimental Parameters

We use the Text-GCN parameter settings to determine our model's parameter settings. The channel number is 200, the batch size is 64, and the embedding size is 300. All models were trained using a 0.001 learning rate. The other parameter settings as shown in Sect. 4.4.

### 4.2.2 Training Details

The model was trained with PyTorch and on an NVIDIA 1660Ti 6 G GPU. For data preprocessing, we referred to the processing method of Text-GCN. For the MR dataset, due to the short text length, we did not remove any words. For other datasets, we only kept words with a word frequency greater than 5 and remove stop words. For all datasets, we did lowercase conversion.

We initialized the word embedding layer during training using Glove-300<sup>4</sup> [41] pre-training word vectors, and we allowed the parameters of the word embedding layer to follow

<sup>1</sup> <http://www.cs.cornell.edu/people/pabo/movie-review-data/>.

<sup>2</sup> <https://www.cs.umb.edu/smimarog/textmining/datasets/>.

<sup>3</sup> <http://disi.unitn.it/moschitti/corpora.htm>.

<sup>4</sup> <http://nlp.stanford.edu/data/glove.6B.zip>.

the model training. Adam [42] optimizer is utilized to train our network. If the correctness of the testing set does not improve after 1000 iterations, training will be stopped. To avoid overfitting the network, we utilized batch-normalization [43] and dropout, where the dropout rate is set to 0.5.

### 4.3 Comparison Models and Experimental Results

#### 4.3.1 Comparison Models

We evaluated the proposed model to the recent models. The following information provides a full comparison of the models:

**CNN** [10]: A convolutional neural network for text classification proposed by Kim. It uses a single-layer multi-channel CNN for feature extraction.

**LSTM** [44]: A long short-term memory network for text classification proposed by Huang et al. The final text representation is based on the LSTM's last step state.

**Bi-LSTM**: Similar to LSTM. However, it uses LSTM for feature extraction in the past and future directions of the text.

**fastText** [45]: A simple and effective text classification model proposed by Joulin et al. It uses the average value of word vectors after word embedding as the final text representation.

**SWEM** [46]: A simple word embedding model proposed by Shen et al. It uses a simple pooling strategy to deal with word embedding.

**LEAM** [47]: A tag embedding attention model proposed by Wang et al. It embeds words and tags into a single space in order to categorize text.

**Transformer** [14]: A normal transformer for text classification.

**GTG** [16]: A transformer and graph convolutional network proposed by Liu et al. It improves the semantic accuracy of word node features.

**Graph-CNN-C** [36]: A GCN model proposed by Defferrard et al. to perform convolution operations on the word embedding similarity map. chebyshev filter is utilized.

**Graph-CNN-S** [48]: Similar to Graph-CNN-C, but a spline filter is employed.

**Graph-CNN-F** [49]: Similar to Graph-CNN-C, but a fourier filter is employed.

**Text-Leve-GNN** [39]: A new text classification model based on GNN proposed by Huang et al. It constructs a graph structure for each text and uses a globally shared edge weight matrix.

**Text-GCN** [18]: A text classification model based on GCN proposed by Yao et al. The problem of text classification is changed into a task of node classification by creating a large graph of the entire corpus.

**Tensor-GCN** [50]: Similar to Text-GCN, but it constructs three graphs of the corpus from three perspectives and then integrates the three graphs to obtain the final node representation.

**GC-GCN-BERT** [51]: A GCN network combined with a gating mechanism proposed by Gao et al. and initialized word vectors using BERT.

**MP-GCN** [52]: A GCN text classification network proposed by Zhao et al., which adds a multi-head pooling module to GCN.

**InducT-GCN** [53]: A novel inductive graph-based text classification framework: inductive graph convolutional networks for text classification.

**Table 2** Accuracy of every comparison model across four datasets

Model	MR	R8	R52	Ohsumed
CNN	77.75 ± 0.72	95.71 ± 0.52	87.59 ± 0.48	58.44 ± 1.06
LSTM	77.33 ± 0.89	96.09 ± 0.19	90.48 ± 0.86	51.10 ± 1.50
Bi-LSTM	77.68 ± 0.86	96.31 ± 0.33	90.54 ± 0.91	49.27 ± 1.07
fastText	75.14 ± 0.20	96.13 ± 0.21	92.81 ± 0.09	57.70 ± 0.49
SWEM	76.65 ± 0.63	95.32 ± 0.26	92.94 ± 0.24	63.12 ± 0.55
LEAM	76.95 ± 0.45	93.31 ± 0.24	91.84 ± 0.23	58.58 ± 0.79
Transformer	76.56 ± 0.65	96.47 ± 1.32	92.12 ± 1.12	52.31 ± 0.94
GTG	77.24 ± 0.30	97.22 ± 0.10	94.46 ± 0.08	69.72 ± 0.13
Graph-CNN-C	77.22 ± 0.27	96.99 ± 0.12	92.75 ± 0.22	63.86 ± 0.53
Graph-CNN-S	76.99 ± 0.14	96.80 ± 0.20	92.74 ± 0.24	62.82 ± 0.37
Graph-CNN-F	76.74 ± 0.21	96.89 ± 0.06	93.20 ± 0.04	63.04 ± 0.77
Text-Leve-GNN	–	97.80 ± 0.20	94.60 ± 0.30	69.40 ± 0.60
Text-GCN	76.74 ± 0.20	97.07 ± 0.10	93.56 ± 0.18	68.36 ± 0.56
Tensor-GCN	77.91 ± 0.07	98.04 ± 0.08	<b>95.05 ± 0.11</b>	<b>70.11 ± 0.24</b>
GC-GCN-BERT	76.25 ± 0.29	97.64 ± 0.05	94.61 ± 0.14	69.53 ± 0.21
InducT-GCN	77.16 ± 0.21	<b>98.07 ± 0.13</b>	94.89 ± 0.27	69.32 ± 0.18
MP-GCN	<b>77.92 ± 0.10</b>	97.85 ± 0.16	94.54 ± 0.08	<b>70.27 ± 0.12*</b>
Ours	<b>78.10 ± 0.34*</b>	<b>98.54 ± 0.21*</b>	<b>95.95 ± 0.24*</b>	69.21 ± 0.54

For each dataset, we perform 10 runs to determine the average accuracy. The result is the average accuracy ± standard deviation. The highest accuracy of each dataset is marked in bold and \*, and the second place is only marked in bold

### 4.3.2 Experimental results

Table 2 contains the outcomes of the experimental comparison. It is evident that the proposed model has delivered good results on four benchmark datasets. On the MR, R8, and R52 datasets, our model had the highest classification accuracy, while it placed third on the Ohsumed dataset.

In detail, compared with classic deep learning text classification models such as CNN, LSTM, and Bi-LSTM, our model produced the best outcomes on four datasets, with an increase of 0.35% (MR), 2.23% (R8), 5.41% (R52) and 10.77% (Ohsumed), respectively. This proves the effectiveness of graph convolution in text feature extraction, and it can extract more effective semantic features than these classical deep learning models. Compared with simple word embedding models such as fastText, SWEM, and LEAM, our model continued to deliver the best outcomes on four datasets, with an increase of 1.15% (MR), 2.41% (R8), 3.01% (R52) and 6.09% (Ohsumed), respectively. Proposed model also shows better results when compared with transformer-based model. Compared to GTG, our model improves 0.86% (MR), 1.32% (R8), and 1.49% (R52) on the three datasets, and reduces 0.51% only on the Ohsumed dataset. This shows that graph convolution can extract deeper semantic features based on original word embedding, thus effectively improving the classification performance. Compared with Text-GCN, our model has achieved stable improvement on four datasets (increased by 1.36% on MR, increased by 1.47% on R8, increased by 2.39% on R52, increased by 0.85% on R52). This shows that it is more effective to construct a single graph for each sentence than Text-GCN, which constructs a large graph for the whole corpus.

Compared with Text-Level-GNN with the same independent composition, our model has achieved stable improvement on R8 and R52 datasets (increased by 0.74% on R8, increased by 1.35% on R52), and lagged by 0.19% in Ohsumed dataset. Compared with InducT-GCN, our model also has achieved improvement on three datasets (increased by 0.94% on MR, increased by 0.47% on R8, increased by 1.06% on R52), and lagged by 0.11% in Ohsumed dataset. Compared with other GCN models, the comparison results are similar to those in InducT-GCN, it has a stable improvement in MR, R8 and R52 datasets, but lagged in the Ohsumed dataset. This shows that our model has a better classification effect for short and medium text, but the effect is relatively weak when dealing with long text (like the Ohsumed dataset). The reason is that the context information of long text will be richer, and our model is set to a fixed value of 1 when the adjacency matrix of the graph convolution is initialized, and the rest of the graph convolution models are calculated by the PMI algorithm. So they will contain richer weight information, and this advantage will be amplified when the text is longer. For example, this is equivalent to our model not using pre-trained word vectors, while the rest of the models do, even though the weights are changed during post-training.

Generally speaking, compared with the traditional non-GNN models, GNN models have shown great advantages in the four datasets. Compared with the other GNN models, our model is more advantageous in short and medium text.

## 4.4 Hyperparameter Research

### 4.4.1 Influence of Network Depth

The influence of network depth on classification accuracy is shown in Fig. 5. On short datasets like MR, the model has the best effect when the depth is 3. On datasets such as R8, R52 and Ohsumed, the best results are achieved when the depth is 5. This validates our assumption that different lengths of text require different network depths. Moreover, the performance of both long and short data sets degrades as network depth continues to increase after the best results have been achieved. The reason is that deepening layers introduce unnecessary noise.

### 4.4.2 Influence of Window Size

The influence of window size on classification accuracy is shown in Fig. 6. It is evident that when the window is relatively small (The window size is 3 or 4), the difference in classification accuracy is relatively small. However, once the window is too large, the classification accuracy will decrease significantly. The reason is that the initial window affects the initial node neighbor distance. When the window is small, the network continuously increases the distance of the captured neighbor nodes as the depth increases. But once the initial window is too large, the network will be unable to extract the information of short-distance nodes (i.e. small-scale features). When deepening the network depth, it is easy to cause the extracted distance range to exceed the sentence length, which increases noise and has a negative impact on classification accuracy.

## 4.5 Exploring the Effectiveness of Attention

To investigate the model's attention mechanism's efficacy, we conduct ablation research on the attention mechanism. The outcomes are displayed in Table 3. Compared with MaxPool

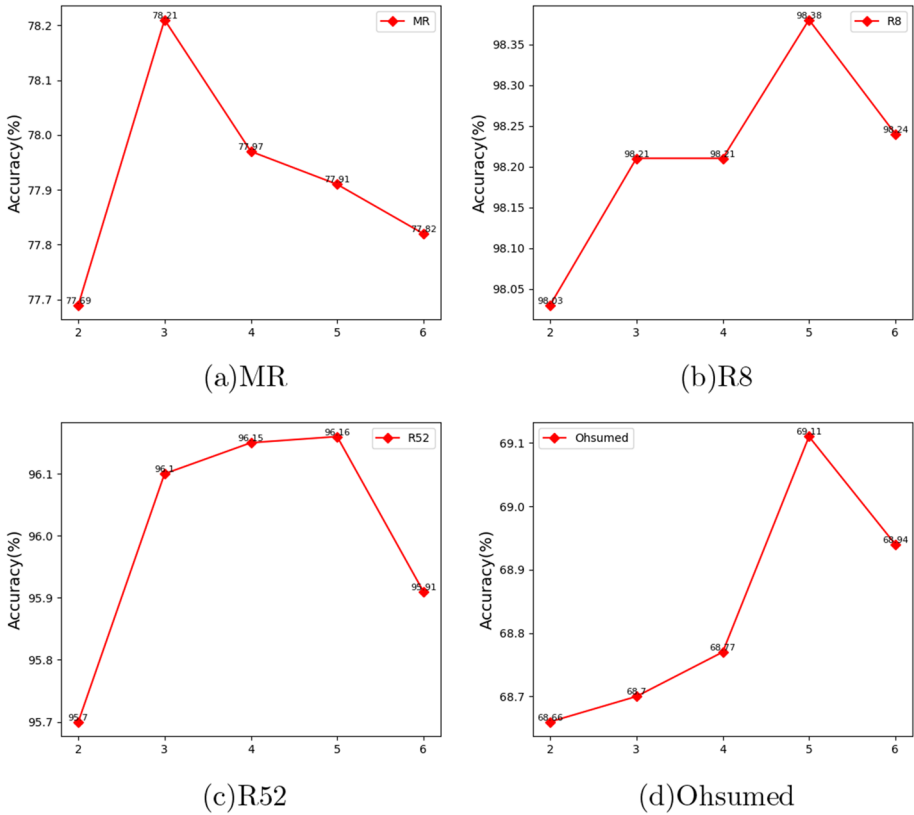


Fig. 5 The effect of network depth on classification accuracy

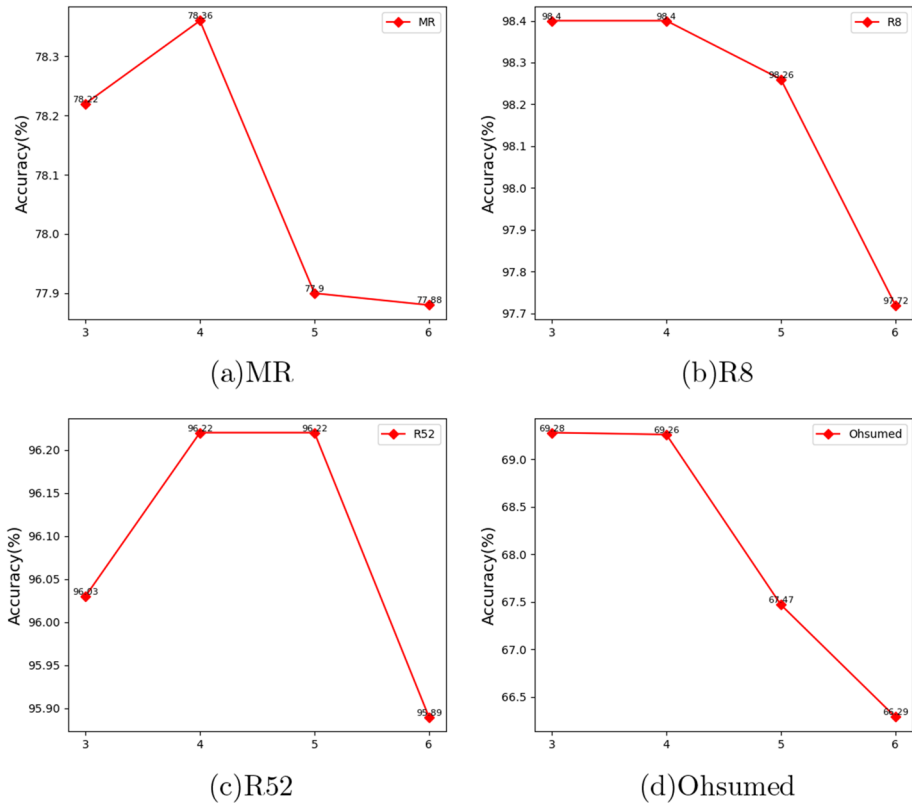
Table 3 Classification accuracy of different pooling methods

Dataset	MR	R8
DC-GCN+MaxPool	78.25	98.26
DC-GCN+MeanPool	78.26	97.99
DC-GCN+Attention	78.53	98.36

and MeanPool, the Attention mechanism has improved the classification accuracy to varying degrees.

### 5 Conclusion

In this research, a densely connected GCN network with attention is proposed for text classification. It starts by creating a unique graph for each text. Then add these independent graphs to a GCN network with dense connections. The GCN network’s high degree of connectivity allows the model to adaptably extract text features at various scales. The attention module receives the extracted features and distinguishes the importance of each aspect when



**Fig. 6** The effect of window size on classification accuracy

aggregating the features, thus improving the performance of the model. Experimental validation uses four benchmark datasets. The experiment results indicate that our model is highly competitive when compared to other advanced methods, particularly for short and medium texts.

Naturally, this is only a preliminary investigation into the use of graph neural networks for text classification. In the future, we will study the application of graph neural networks in text classification in more detail. For example, it is expected that the field of text classification will use the currently popular graph attention networks and gated graph convolution to significantly improve classification performance.

**Acknowledgements** This paper was supported by the Open Fund of Hubei Key Laboratory of Intelligent Geo-Information Processing (ZRIGIP-201801).

**Author Contributions** Y.P.: Conceptualization, Methodology, Software, Writing—Original Draft, Writing—Review and Editing, Visualization. W.W.: Conceptualization, Methodology, Software, Writing—Original Draft, Writing—Review and Editing. J.R.: Supervision, Funding acquisition. X.Y.: Writing—Review and Editing.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Souza E, Santos D, Oliveira G, Silva A, Oliveira AL (2020) Swarm optimization clustering methods for opinion mining. *Nat Comput* 19(3):547–575
2. Shrivras AK, Dewangan AK, Ghosh S, Singh D (2021) Development of proposed ensemble model for spam e-mail classification. *Inf Technol Control* 50(3)
3. He C, Hu Y, Zhou A, Tan Z, Zhang C, Ge B (2020) A web news classification method: fusion noise filtering and convolutional neural network. In: 2020 2nd symposium on signal processing systems, pp 80–85
4. Minaee S, Kalchbrenner N, Cambria E, Nikzad N, Chenaghlu M, Gao J (2020) Deep learning based text classification: a comprehensive review. *arXiv preprint arXiv:2004.03705*
5. Zhou Z, Qin J, Xiang X, Tan Y, Liu Q, Xiong NN (2020) News text topic clustering optimized method based on TF-IDF algorithm on spark. *Comput Mater Contin* 62(1):217–231
6. García M, Maldonado S, Vairetti C (2021) Efficient n-gram construction for text categorization using feature selection techniques. *Intell Data Anal* 25(3):509–525
7. Aksoy G, Karabatak M (2019) Performance comparison of new fast weighted Naïve Bayes classifier with other Bayes classifiers. In: 2019 7th international symposium on digital forensics and security (ISDFS). IEEE, pp 1–5
8. Guo H, Wang W (2019) Granular support vector machine: a review. *Artif Intell Rev* 51(1):19–32
9. Le L, Xie Y, Raghavan VV (2018) Deep similarity-enhanced k nearest neighbors. In: 2018 IEEE international conference on big data (big data). IEEE, pp 2643–2650
10. Kim Y (2014) Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*
11. Conneau A, Schwenk H, Barrault L, Lecun Y (2016) Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*
12. Chang C, Masterson M (2020) Using word order in political text classification with long short-term memory models. *Polit Anal* 28(3):395–411
13. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*
14. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. *Adv Neural Inf Process Syst* 30
15. Shi Y, Zhang X, Yu N (2023) Pl-transformer: a pos-aware and layer ensemble transformer for text classification. *Neural Comput Appl* 35(2):1971–1982
16. Liu B, Guan W, Yang C, Fang Z, Lu Z (2023) Transformer and graph convolutional network for text classification. *Int J Comput Intell Syst* 16(1):161
17. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. *IEEE Trans Neural Netw* 20(1):61–80
18. Yao L, Mao C, Luo Y (2019) Graph convolutional networks for text classification. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 7370–7377
19. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*
20. Yang C, Wang R, Yao S, Liu S, Abdelzaher T (2020) Revisiting “over-smoothing” in deep gens. *arXiv preprint arXiv:2003.13663*
21. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
22. Hu D (2019) An introductory survey on attention mechanisms in nlp problems. In: Proceedings of SAI intelligent systems conference. Springer, pp 432–448
23. Kowsari K, Jafari Meimandi K, Heidarysafa M, Mendu S, Barnes L, Brown D (2019) Text classification algorithms: a survey. *Information* 10(4):150

24. Chen Y, Xu L, Liu K, Zeng D, Zhao J (2015) Event extraction via dynamic multi-pooling convolutional neural networks. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: long papers), pp 167–176
25. Xu J, Cai Y, Wu X, Lei X, Huang Q, Leung H-F, Li Q (2020) Incorporating context-relevant concepts into convolutional neural networks for short text classification. *Neurocomputing* 386:42–53
26. Johnson R, Zhang T (2017) Deep pyramid convolutional neural networks for text categorization. In: Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: long papers), pp 562–570
27. Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. *Adv Neural Inf Process Syst* 28:649–657
28. Zhang X, LeCun Y (2015) Text understanding from scratch. arXiv preprint [arXiv:1502.01710](https://arxiv.org/abs/1502.01710)
29. Tai KS, Socher R, Manning CD (2015) Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint [arXiv:1503.00075](https://arxiv.org/abs/1503.00075)
30. Zhang T, Huang M, Zhao L (2018) Learning structured representation for text classification via reinforcement learning. In: Thirty-second AAAI conference on artificial intelligence
31. Xu C, Huang W, Wang H, Wang G, Liu T-Y (2019) Modeling local dependence in natural language with multi-channel recurrent neural networks. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 5525–5532
32. Lai S, Xu L, Liu K, Zhao J (2015) Recurrent convolutional neural networks for text classification. In: Twenty-ninth AAAI conference on artificial intelligence
33. Zhou C, Sun C, Liu Z, Lau F (2015) A c-lstm neural network for text classification. arXiv preprint [arXiv:1511.08630](https://arxiv.org/abs/1511.08630)
34. Liu G, Guo J (2019) Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing* 337:325–338
35. Yan S, Xiong Y, Lin D (2018) Spatial temporal graph convolutional networks for skeleton-based action recognition. arXiv preprint [arXiv:1801.07455](https://arxiv.org/abs/1801.07455)
36. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. arXiv preprint [arXiv:1606.09375](https://arxiv.org/abs/1606.09375)
37. Peng H, Li J, He Y, Liu Y, Bao M, Wang L, Song Y, Yang Q (2018) Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In: Proceedings of the 2018 world wide web conference, pp 1063–1072
38. Liu X, You X, Zhang X, Wu J, Lv P (2020) Tensor graph convolutional networks for text classification. In: AAAI, pp 8409–8416
39. Huang L, Ma D, Li S, Zhang X, Wang H (2019) Text level graph neural network for text classification. arXiv preprint [arXiv:1910.02356](https://arxiv.org/abs/1910.02356)
40. Pang B, Lee L (2005) Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. arXiv preprint [cs/0506075](https://arxiv.org/abs/cs/0506075)
41. Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543
42. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)
43. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint [arXiv:1502.03167](https://arxiv.org/abs/1502.03167)
44. Liu P, Qiu X, Huang X (2016) Recurrent neural network for text classification with multi-task learning. arXiv preprint [arXiv:1605.05101](https://arxiv.org/abs/1605.05101) (2016)
45. Joulin A, Grave E, Bojanowski P, Mikolov T (2016) Bag of tricks for efficient text classification. arXiv preprint [arXiv:1607.01759](https://arxiv.org/abs/1607.01759)
46. Shen D, Wang G, Wang W, Min MR, Su Q, Zhang Y, Li C, Heno R, Carin L (2018) Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. arXiv preprint [arXiv:1805.09843](https://arxiv.org/abs/1805.09843)
47. Wang G, Li C, Wang W, Zhang Y, Shen D, Zhang X, Heno R, Carin L (2018) Joint embedding of words and labels for text classification. arXiv preprint [arXiv:1805.04174](https://arxiv.org/abs/1805.04174)
48. Bruna J, Zaremba W, Szlam A, LeCun Y (2013) Spectral networks and locally connected networks on graphs. arXiv preprint [arXiv:1312.6203](https://arxiv.org/abs/1312.6203)
49. Henaff M, Bruna J, LeCun Y (2015) Deep convolutional networks on graph-structured data. arXiv preprint [arXiv:1506.05163](https://arxiv.org/abs/1506.05163)
50. Liu X, You X, Zhang X, Wu J, Lv P (2020) Tensor graph convolutional networks for text classification. World Wide Web, Geneva



51. Gao W, Huang H (2021) A gating context-aware text classification model with bert and graph convolutional networks. *J Intell Fuzzy Syst* 40(3):4331–4343
52. Zhao H, Xie J, Wang H (2022) Graph convolutional network based on multi-head pooling for short text classification. *IEEE Access* 10:11947–11956. <https://doi.org/10.1109/ACCESS.2022.3146303>
53. Wang K, Han SC, Poon J (2022) Induct-gcn: inductive graph convolutional networks for text classification. In: 2022 26th international conference on pattern recognition (ICPR). IEEE, pp 1243–1249

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.