Complex & Intelligent Systems (2024) 10:2627–2638
https://doi.org/10.1007/s40747-023-01289-9

ORIGINAL ARTICLE

Bert-based graph unlinked embedding for sentiment analysis

Youkai Jin[1] · **Anping Zhao**[1]

Received: 16 July 2023 / Accepted: 4 November 2023 / Published online: 8 December 2023

Abstract
Numerous graph neural network (GNN) models have been used for sentiment analysis in recent years. Nevertheless, addressing the issue of over-smoothing in GNNs for node representation and finding more effective ways to learn both global and local information within the graph structure, while improving model efficiency for scalability to large text sentiment corpora, remains a challenge. To tackle these issues, we propose a novel Bert-based unlinked graph embedding (BUGE) model for sentiment analysis. Initially, the model constructs a comprehensive text sentiment heterogeneous graph that more effectively captures global co-occurrence information between words. Next, by using specific sampling strategies, it efficiently preserves both global and local information within the graph structure, enabling nodes to receive more feature information. During the representation learning process, BUGE relies solely on attention mechanisms, without using graph convolutions or aggregation operators, thus avoiding the over-smoothing problem associated with node aggregation. This enhances model training efficiency and reduces memory storage requirements. Extensive experimental results and evaluations demonstrate that the adopted Bert-based unlinked graph embedding method is highly effective for sentiment analysis, especially when applied to large text sentiment corpora.

Keywords Sentiment analysis · GNNs · Bert · Graph unlinked embedding

Introduction

Sentiment analysis is a prominent research area in natural language processing, involving the automatic extraction of emotional tendencies from text and the determination of people's emotional attitudes towards various trending topics [1]. Text sentiment analysis plays a vital role in various fields, including government management, movie recommendations, public opinion analysis, and assisting researchers in making more informed strategic decisions [2].

In recent years, graph neural network (GNN) models have garnered significant attention for their effectiveness in modeling graph structures, particularly in applications like text sentiment analysis. Yao et al. introduced GCN (graph convolutional network) for heterogeneous text graphs, achieving promising results [3]. Niu et al. proposed a syntax-enhanced graph neural network model for sentiment analysis, further enhancing model performance [4]. However, GNN-based models still face challenges when dealing with text sentiment graphs [5]. Existing GNN models heavily rely on edge connections between graph nodes for representation learning. When constructing text sentiment graphs, the sheer volume of edges created results in high memory consumption. Additionally, due to the presence of edges, as GNN models undergo continuous training, the node representations obtained by such deep models tend to become overly smoothed and indistinguishable [6]. GNN models have demonstrated their ability to effectively preserve the global information of graph structures. However, there are challenges in handling local information among nodes. Therefore, when dealing with sentiment graphs in text processing, preserving the heterogeneity of text, improving model efficiency for effective scalability to large text sentiment corpora, and effectively retaining both global and local information within the graph structure have become significant challenges.

This work was supported by the National Natural Science Foundation of China (Grant No. 62277040).

✉ Anping Zhao
apzhao@wzu.edu.cn

Youkai Jin
jyk0620@126.com

[1] The Zhejiang Key Laboratory of Intelligent Informatics for Safety and Emergency, Wenzhou University, Wenzhou, Zhejiang 325035, China

We distinguish ourselves from existing GNN models by introducing a novel Bert-based unlinked graph embedding (BUGE) model for text sentiment analysis. We utilize a small batch of linkless subgraph decomposition for input text sentiment graphs, breaking down large heterogeneous text sentiment graphs into several subgraphs without edge connections. This effectively reduces model memory usage and extends its applicability to large text corpora. By employing specific sampling strategies during the sampling process, we can efficiently preserve both global and local information within the graph structure, enabling nodes to receive more feature information. In the representation learning process, BUGE relies solely on attention mechanisms [7], without employing graph convolutions or aggregation operators, thereby addressing the issue of node oversmoothing and enhancing model training efficiency.

In this paper, our focus is on simplifying the edge connections between nodes in existing GNN models while retaining the heterogeneity of text and effectively preserving both global and local information within the graph structure. We aim to address the issue of node oversmoothing during model training. To achieve this, we employ specific sampling strategies on large-scale text sentiment heterogeneous graphs, creating unlinked small-batch isomorphic subgraphs. This approach eliminates the dependence on edges between graph nodes and can be effectively scaled to large text sentiment corpora. We demonstrate the effectiveness of this sampling method in current research on text sentiment analysis. Our primary contributions are as follows:

(1) We introduce the BUGE model for sentiment analysis. We process the input text sentiment graph using small-batch unlinked isomorphic subgraph decomposition. This approach retains the heterogeneity of text while reducing the model's storage and computational requirements, making it effectively scalable to large sentiment corpora.
(2) We employ a specific subgraph sampling strategy that preserves local information while retaining the global information of the graph structure. During the representation learning process, the model relies entirely on attention mechanisms, thus addressing the issue of node oversmoothing and enhancing model efficiency.
(3) We conduct experiments on several benchmark datasets, and the results validate the effectiveness of our model.

Related work

Graph based sentiment analysis

Recently, GNN has achieved good results in sentiment analysis [8], attracting the attention of many researchers. Sentiment analysis models the content text as a sentiment graph for feature extraction and then uses GNN for graph embedding. During graph embedding, the GNN model can maintain the global structural information of the sentiment graph and effectively deal with the complex relational structure between text sentiment words. Therefore, GNN is widely used in sentiment analysis tasks.Yao et al. proposed the Text-GCN model for text classification, which constructs a large text heterogeneity graph to describe the word–document and word–word relationships and then uses GCN model to complete text classification [3]. Huang et al. proposed TLGNN, a model that constructs a graph for each input text with global parameter sharing, alleviating the dependency between an individual text and the entire corpus [9]. However, these models require pre-constructed graph structures, which have limitations in practical applications. Therefore, Ding et al. proposed HyperGAT, which effectively captures complex node associations and hyperedge relationships through its graph attention mechanism and multi-layer structure. However, it is constrained by attention distribution when handling local information in the graph structure [10]. Zhu et al. proposed SSGC, where they added a self-loop to the Markov diffusion kernel and proposed a straightforward spectral map convolution. The simple spectral graph convolution used in this context strikes a balance between low-pass and high-pass filter frequency bands, effectively capturing both global and local information of nodes [11]. Zhang et al. proposed TextING, which incorporates a gating mechanism to alleviate the issue of excessive node smoothing. It is an inductive and versatile text classification model that can handle diverse text sentiment analysis tasks. However, it relies heavily on the quality of the text graph [12]. Zhu et al. proposed GL-GCN, a graph convolutional network guided by both global and local dependencies. It employs two GCNs to learn different dependency structures effectively, capturing both global and local contextual information [13]. Yang et al. proposed the CGA2TC model, which employs a contrastive learning approach to enhance model classification performance by using two different views. It should be noted that CGA2TC's use of two text views increases computational overhead and does not scale well to large text corpora graphs [14].

Bert-based graph representation

In recent years, the powerful BERT model has made significant strides in natural language processing [15], prompting researchers to apply it to graph representation learning by combining BERT with GCN. Lu et al. proposed a VGCN-BERT model that combines the pre-trained BERT model with a lexical graph convolutional network to construct a large text heterogeneous graph. The attention mechanism is then used to interact with local and global information, influencing each other to jointly construct classification representations [16].

Yang et al. proposed a BERT-enhanced text network model (BEGNN) that considers both the semantic and structural information of a single text. It constructs a graph structure for each text and combines the graph neural network with BERT to extract features of varying granularity [17]. Lin et al. introduced a BERT-GCN model for text classification, which uses a graph to model the relationships between different samples from the entire corpus to leverage the similarity between labeled and unlabeled documents. It employs GNNs to learn these relationships [18]. Hao et al. introduced a novel defect prediction framework named EDP-BGCNN. This framework effectively harnesses the powerful capabilities of BERT and GCN for code representation and analysis, leading to more accurate defect prediction and enhancing the precision of defect prediction [19]. Numerous studies have demonstrated the effectiveness of Bert-based graph representation in related domains.

Compared to existing research, our work differs in several aspects. When dealing with text sentiment graphs, existing models often need to be more efficient due to the complexity of the constructed text sentiment graphs. As a result, we employ the method of small-batch linkless subgraph decomposition to partition a large text heterogeneous graph into multiple small-batch linkless subgraphs. This allows the model to process the graph without relying on edge connections, significantly enhancing its operational efficiency when applied to large text sentiment corpora. Throughout this process, we use specific sampling strategies to effectively preserve both the global and local information within the graph structure, enabling nodes to receive more feature information. Furthermore, our model relies solely on the attention mechanism, without the utilization of any graph convolutions or aggregation operators. This approach effectively resolves the issue of node oversmoothing present in existing GNN models.

Problem definition

Definition 1 *Text sentiment network*

We construct a text sentiment heterogeneous graph $\mathcal{G} = (\mathcal{W}, \mathcal{S}, \mathcal{X}, \mathcal{Y})$, where $\mathcal{W}$ and $\mathcal{S}$ respectively represent word nodes and sentiment nodes, $\mathcal{X}$ represents edges between word nodes, and $\mathcal{Y}$ represents edges between word nodes and sentiment nodes.

The text sentiment heterogeneous graph links words and sentiment words in the text together, represented as a graph.

Definition 2 *Text graph embedding*

Given an input graph $\mathcal{G}$, we define U as the set of all nodes in the text sentiment graph. The task of text graph embedding is to learn a mapping function $f : u_i \rightarrow u_i \in \mathbb{R}^d$ that embeds the nodes $u \in U_i$ of the text graph network into low-dimensional latent representations $X \in \mathbb{R}^{|U| \times d}$, where $d \leq |U|$. These embeddings capture structural and sentiment information between nodes.

The obtained node embedding vectors can be used as feature inputs and embedded to complete the task of predicting sentiment relationships.

Definition 3 *Node neighbourhood of linkless subgraph*

We calculate the intimacy matrix Γ between nodes. For each target node $u_i \in U$, we define its learning context as set $\zeta_{u_i} = \{u_j | u_j \in U \setminus \{u_i\} \wedge \Gamma(i, j) \geq \theta_i\}$, $\Gamma(i, j)$ measures the closeness score between word node u_i and word node u_j, and θ_i defines the minimum intimacy score threshold for nodes involved in u_i's context.

To find the k nearest neighbor word nodes $u_j \in U$ with the highest intimacy to node u_i according to Γ, we can use ζ_{u_i} to select the top-k intimate nodes of u_i in the graph $\mathcal{G}$. By combining the context ζ_{u_i} of the word node u_i and the node u_i itself, we can form a linkless subgraph g_i. This complete heterogeneous text sentiment graph can be expressed as $\mathcal{G} = g_1, g_2, ..., g_u$.

Through this definition, we can determine the composition of neighbor nodes in the linkless subgraph, including nodes that are close to this node in the original large heterogeneous graph and those that are farther away.

Definition 4 *Sentiment relationship prediction*

For the prediction of sentiment relationship, we predict the sentiment contained in the target text based on the constructed text sentiment graph $\mathcal{G}$. We define the prediction function as $g : (\mathcal{W}, \mathcal{S}, \mathcal{X}, \mathcal{Y}, v_i) \rightarrow Z$ to predict v_i's sentiment relationship where $Z = [Z_1, Z_2, \ldots, Z_i]$ represents the different possible results of the sentiment relationship prediction for v_i.

Proposed method

Overview

This section will explain how the model is utilized for text sentiment classification, as illustrated in Fig. 1. Initially, we introduce the creation of a heterogeneous text graph for sentiment analysis. Subsequently, we describe the sampling method used to generate multiple linkless connected subgraphs from the large text heterogeneous composition. Next, we discuss how node inputs are embedded in these subgraphs, followed by learning the node representation through the graph transformer encoder for classification.

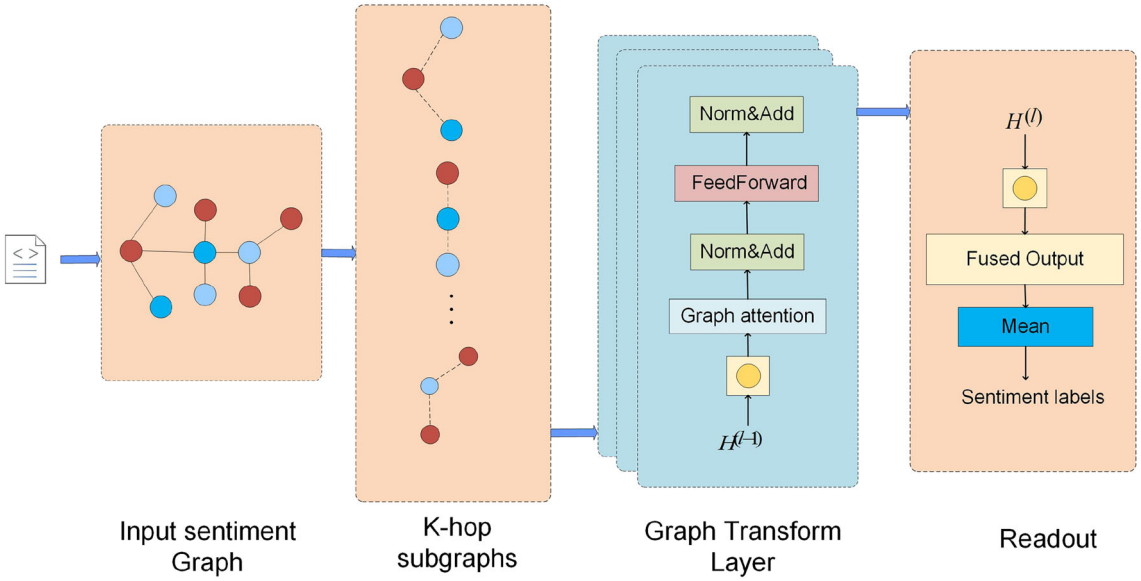

Fig. 1 The Architecture of the BUGE Model. (For example, a text is constructed into a text sentiment graph, and subgraphs are decomposed into several linkless subgraphs. We update the representation of nodes through the graph transformer layer and obtain the text sentiment classification through the readout function)

Text sentiment graph

We aim to create a comprehensive and diverse text graph that includes both word and sentiment nodes to facilitate sentiment analysis, with a focus on capturing global word co-occurrences. To achieve this, we construct a text sentiment heterogeneous graph denoted as $G = (\mathcal{W}, \mathcal{S}, \mathcal{X}, \mathcal{Y})$, following the approach of Text-GCN. We establish edges between nodes to form a large and complex graph for the entire corpus. The weight of the edge between a sentiment node and a word node is determined by the word's inverse document frequency (TF-IDF) in the document. Additionally, to leverage global word co-occurrence information, we collect word co-occurrence statistics by applying a fixed-size sliding window over all documents in the corpus. We calculate the weights of edges between two-word nodes using pointwise mutual information (PMI). Specifically, the weight of an edge between node w_i and node w_j is defined as:

$$\begin{cases} PMI(i,j) & i,j \text{ are words, } PMI(i,j) > 0 \\ TF-IDF(i,j) & i \text{ is sentiment word, } j \text{ is word} \\ 1 & i = j \\ 0 & otherwise \end{cases} \tag{1}$$

The formula for TF-IDF is:

$$PMI\left(w_i, w_j\right) = \log \frac{p_{i,j}}{p_i p_j} = \log \frac{N_{i,j} N}{N_i N_j} \tag{2}$$

where N_i and N_j are the number of sliding windows in the corpus that contain the word w_i and w_j respectively, and $N_{i,j}$ is the number of sliding windows that contain both words w_i and w_j. N is the total number of sliding windows in the corpus.

The formula for TF-IDF is:

$$TF-IDF(w,s) = TF(w,s) * IDF(w) \tag{3}$$

Here, w represents a word, and s represents a sentiment word. $TF(w,s)$ represents the frequency of word w in a sentiment word s, i.e., the number of times w appears in s. $IDF(w)$ represents the inverse document frequency of word w and can be calculated using the following formula:

$$IDF(w) = log(N/(1+n(w))) \tag{4}$$

Here, N represents the total number of sentiment words in the corpus, and $n(w)$ represents the number of sentiment words that contain the word w.

Bert-based graph embeddings learning

We have incorporated the Graph-bert model [20] into text sentiment analysis. It adopts the top-k intimacy sampling approach to calculate the intimacy degree between each node and all other nodes. It then selects the top k nodes with the largest intimacy values as neighbor nodes. It calculates the intimacy matrix Γ between the nodes in the complete graph

using PageRank as follows:

$$\Gamma = \alpha \cdot \left(I - (1-\alpha) \cdot \bar{A} \right)^{-1} \tag{5}$$

Here, α is a factor in the range of [0, 1]. The term $\overline{A} = AD^{-1}$ denotes the column-normalized adjacency matrix, where A is the adjacency matrix of the input graph, and D is the diagonal matrix corresponding to its diagonal.

According to the calculated intimacy matrix Γ, for each sentiment target node $s_i \in S$, we define its learning context as the set:

$$\zeta_{(S_i)} = \left\{ w_j \middle| w_j \in V \setminus \{w_i + S_i\} \wedge \Gamma(i, j) \geq \theta_i \right\} \tag{6}$$

Here, θ_i is the minimum intimacy score threshold for nodes to be involved in s_i's context.

For each sentiment target node $s_i \in S$, we select the closest k neighbor nodes $w_i \in W$ to sample the context subgraph ζ_{s_i} of size k.

The nodes in the text sentiment analysis graph are updated for classification based on the sampled linkless connected subgraphs. Node feature vectors are obtained through embeddings of model input nodes.

Raw Feature Vector Embedding: Raw Feature Vector Embedding can capture the sentiment text and word types in the text graph. For each subgraph node $W_j \in V_i$, we embed the raw feature vector into a feature space, where the raw feature vector is denoted as y_j.

$$q_j^{(x)} = \text{Embed}(y_j) \in \mathbb{R}^{d_h \times 1} \tag{7}$$

Here, the definition of the $Embed(\cdot)$ function can be implemented using different models, and here we use BERT.

Relative Positional Embedding: We define the position of word nodes as $P(v_i)$. By default, p_{v_i} is set to 0, and nodes closer to v_i have smaller position indices. For node v_j, we can also extract its intimacy-based relative positional embedding using the $P\text{-}Embed(\cdot)$ function defined above as follows:

$$q_j^{(r)} = \text{P-Embed}(P(v_i)) \in \mathbb{R}^{d_h \times 1} \tag{8}$$

Then, we apply the graph transformer method to the graph structure data. After calculating the two embedding vectors defined above, we aggregate them together as input to the encoder.

We organize all the input vectors in the subgraph g_i into a matrix $H^{(0)} = \left[h_i^{(0)}, h_{i,1}^{(0)}, \ldots, h_{i,k}^{(0)} \right]^T \in \mathbb{R}^{(k+1) \times d_h}$, and then the representation of the node is gradually updated through the multi-layer attention operation:

$$\begin{aligned} H^{(l)} &= G\text{-Transformer}(H^{(l-1)}) \\ &= \text{softmax}\left(\frac{QK^T}{\sqrt{d_h}} \right) V + G\text{-Res}(H^{l-1}, X_i) \end{aligned} \tag{9}$$

where

$$\begin{cases} Q = H^{(l-1)} W_Q^{(l)} \\ K = H^{(l-1)} W_K^{(l)} \\ V = H^{(l-1)} W_V^{(l)} \end{cases} \tag{10}$$

In the above equation, each graph transformer layer contains three trainable matrices: $W_Q^{(l)}, W_K^{(l)}, W_V^{(l)} \in \mathbb{R}^{(d_h \times d_h)}$, and queries Q, keys K, and values V are generated by multiplying the input correspondingly. G-Res [21] refers to a residual network for solving the over-smoothing problem of GNNs.

The representation fusion layer averages the output embeddings of the D-th encoder layer to obtain z_i as the final representation of the target node v_i:

$$z_i = \text{Fusion}(H^{(D)}) \tag{11}$$

Text sentiment node classification

After learning the node representations, we classify the nodes. The representation fusion layer averages the output embedding of the D-th encoder layer to obtain z_i as the final representation of the target node v_i, which is then fed into a softmax classifier:

$$z_i = \text{softmax}\left(\text{average}\left(H^{(D)} \right) \right) \in R^{d_h \times 1} \tag{12}$$

In comparison with the real labels of nodes, we define the cross-entropy loss function as follows:

$$\mathcal{L} = -\sum_{n \in \mathcal{T}} \sum_{f=1}^{d_y} y_n(f) \log z_n(f) \tag{13}$$

Here, $n \in \mathcal{T}$ denotes the target word/sentiment word node in the training set, d_y is the label vector dimension, and y_n denotes the ground truth label vector of node n.

Through the joint training of the fully connected layer constructed above and the model, we can determine the label type of the node.

Experimental results and evaluation

In this section, we will evaluate the performance of the proposed model in text sentiment analysis. We will compare our

Table 1 Summary statistics of datasets

Datasets	Training	Test	Nodes	Classes	Length
MR	8530	2132	29,426	2	20.4
IMDB	25,000	25,000	90,653	2	232.8
SST-5	9484	2371	19,849	5	20.1

proposed model with classic text sentiment analysis baseline methods, demonstrate the superiority of our model in text sentiment analysis, and document and analyze the experimental results.

Datasets

We conducted extensive experiments on three benchmark datasets:

MR (Movie Review): This dataset is used for binary sentiment classification of movie reviews, each consisting of a single sentence [22]. It comprises 10,662 samples, evenly split between 5331 positive and 5331 negative reviews [23].

IMDB (IMDB Movie Reviews): This large text sentiment corpus contains more data than previous benchmarks, consisting of 50,000 reviews from the Internet Movie Database, labeled as positive or negative [24]. It encompasses 25,000 positive and 25,000 negative sample reviews.

SST-5 (The Stanford Sentiment Treebank): Stanford University released this sentiment analysis dataset, which includes five labels: very positive, positive, neutral, negative, and very negative. This dataset provides a clearer distinction between emotions [25]. Detailed statistics for these datasets are listed in Table 1.

Experimental setup

For our experiments, we initialized the embeddings with pre-trained 300-dimensional Glove vectors [26]. We trained a two-layer graph transformer with a hidden size of 32 and 6 attention heads. We used Adam as the optimizer [27] with an initial learning rate of 0.001, which was decayed by a factor of $1e-5$. We set the number of training epochs to 50.

Evaluation metrics

In our experiments, we adopted accuracy, precision, recall, and $F1$-score as performance metrics. accuracy measures the model's ability to correctly classify samples, precision focuses on the model's accuracy in the positive class, recall assesses the model's capability to identify positive class samples, and $F1$-score provides a comprehensive evaluation by considering both Precision and Recall. These metrics help us comprehensively assess the model's performance.

Accuracy: In binary classification, accuracy is calculated as the proportion of correctly predicted positive and negative samples to the total number of samples. In multi-class classification, accuracy is calculated by dividing the sum of correctly predicted samples by the total number of samples.

$$\mathrm{Acc} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{14}$$

where TP = true positive, FP = false positive, TN = true negative, and FN = false negative.

Precision: In binary classification, Precision represents the proportion of correctly predicted positive samples to all samples that were predicted as positive. In multi-class classification, Precision is calculated separately for each label, and then a weighted average is computed to account for class imbalances.

$$\mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{15}$$

Recall: In binary classification, Recall, also known as True Positive Rate, measures the proportion of positive samples that were correctly predicted out of the total number of actual positive samples in the dataset. In multi-class classification, Recall is calculated for each label by dividing the number of correctly predicted samples in that label by the total number of actual samples in that label. A weighted average is then computed to account for class imbalances across labels.

$$\mathrm{Re} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{16}$$

F1-score: The $F1$-score is the harmonic mean of Precision and Recall. The $F1$-score is calculated using the following formula:

$$\frac{2}{F_1} = \frac{1}{\mathrm{Pr}} + \frac{1}{\mathrm{Re}} \tag{17}$$

Methods for comparison

We adopt the current popular text sentiment analysis models to conduct a performance comparison evaluation of our proposed model:

Text-GCN [3]: Constructs a heterogeneous graph based on text and words, enabling semi-supervised text classification using Graph Convolutional Networks.

TLGCN [9]: Builds a graph for each input text with global parameter sharing, alleviating the dependency between a single text and the entire corpus.

SSLGNN [28]: Introduces a new sparse structure learning model based on Graph Neural Networks.

HyperGAT [10]: it is a variant of Graph Neural Network designed to handle hypergraph data.

TensorGCN [29]: Utilizes a model for multi-angle mapping and fusion.

TextING [12]: Constructs a separate graph for each text and updates nodes through a message propagation mechanism.

SSGC [11]: Adds a self-loop to the Markov diffusion kernel and proposes a simple spectral map convolution with a softmax classifier after a linear layer.

CGA2TC [14]: Presents a novel framework of contrastive graph convolutional networks with adaptive augmentation.

GFN [30]: Introduces a unified graph fusion network that transforms external knowledge into structural information.

Results

Results analysis

We employ multiple metrics to present the experimental results of various sentiment analysis classification methods on our dataset. The experimental outcomes are detailed in Tables 2, 3, and 4. Our experiments demonstrate that our proposed model performs exceptionally well on benchmark datasets. Notably, IMDB represents a large sentiment text corpus with lengthier content in the text and a more extensive sample size, while each item in the MR dataset consists of relatively shorter text content. SST5 is a multi-category dataset. It's worth noting that graph-based methods consistently excel across different datasets. Our model exhibits significant improvements on the IMDB dataset, primarily owing to the considerably longer average text length in IMDB compared to other datasets. When constructing text sentiment graphs, longer texts can establish more sentiment word connections transmitted through nodes, which enhances the ability to capture the relationship between target nodes and sentiment words. When combined with BUGE, this leads to superior performance. This might elucidate why the improvement is more pronounced on large text sentiment analysis datasets than on smaller text sentiment corpora like MR and SST5, which feature shorter texts. The constraints of the graph structure during text graph construction result in only slight performance improvements for datasets with shorter texts.

Text-GCN and TLGCN use graph neural networks to train models, but both struggle to effectively preserve the heterogeneity of text data and face scalability challenges when applied to large-scale text sentiment corpora. TextING may not be effective in extracting text features while ignoring word order, especially in sentiment classification tasks. HyperGAT neglects the capture of local information in the graph structure, leading to the model's inability to capture label correlations, thereby reducing its performance, especially in multi-label tasks.GFN introduces an innovative approach to construct a text sentiment graph, but like HyperGAT, it also fails to adequately capture local information within the graph structure. TensorGCN relies on pre-trained word embeddings and tends to perform well with shorter texts but fares poorly with longer ones. SSGC incorporates double-layer attention mechanisms, which prove advantageous for learning tasks involving lengthy texts. CGA2TC uses the multi-view contrasting method to enhance the classification performance of the model. However, the use of multi-view methods typically increases computational complexity due to the need to handle features from multiple views. This results in longer training times and higher computational resource requirements, causing the model to perform poorly on large text sentiment corpora. In addition, the models mentioned above all face the issue of node smoothing during the model

Table 2 Comparison of performance on different sentiment analysis approaches on the MR dataset

Model	MR			
	Accuracy	Precision	Recall	$F1$-score
TextGCN	0.7674	0.7548	0.7546	0.7547
TLGCN	0.7547	0.7412	0.7429	0.7420
SSLGNN	0.7974	0.7912	0.7896	0.7904
HyperGAT	0.7652	0.7598	0.7612	0.7605
TensorGCN	0.7789	0.7686	0.7627	0.7656
TextING	0.7982	0.7653	0.7622	0.7637
SSGC	0.7675	0.7675	0.7661	0.7668
CGA2TC	0.7780	0.7727	0.7756	0.7741
GFN	0.7807	0.7806	0.7806	0.7784
BUGE	**0.8044**	**0.7972**	**0.7973**	**0.7973**

Bold values represent the best experimental performance results

Table 3 Comparison of performance on different sentiment analysis approaches on the IMDB dataset

Model	IMDB			
	Accuracy	Precision	Recall	$F1$-score
TextGCN	–	–	–	–
TLGCN	–	–	–	–
SSLGNN	–	–	–	–
HyperGAT	0.8346	0.8284	0.8265	0.8275
TensorGCN	0.8541	0.8343	0.8411	0.8377
TextING	0.8623	0.8597	0.8610	0.8603
SSGC	0.8726	0.8637	0.8692	0.8664
CGA2TC	0.8631	0.8628	0.8626	0.8627
GFN	0.8562	0.8462	0.8512	0.8487
BUGE	**0.8957**	**0.8856**	**0.8906**	**0.8881**

The data is sourced from existing research reports, and for scores that have not been reported in the existing studies, we indicate them with a "–" in the table

Bold values represent the best experimental performance results

Table 4 Comparison of performance on different sentiment analysis approaches on the SST-5 dataset

Model	SST-5			
	Accuracy	Precision	Recall	$F1$-score
TextGCN	0.4075	0.4043	0.4059	0.4051
TLGCN	0.4303	0.4292	0.4143	0.4217
SSLGNN	0.4398	0.4363	0.4318	0.4340
HyperGAT	0.4135	0.4032	0.4053	0.4043
TensorGCN	0.4368	0.4361	0.4364	0.4362
TextING	0.4289	0.4182	0.4235	0.4208
SSGC	0.4224	0.4024	0.4047	0.4036
CGA2TC	0.4498	0.4435	0.4466	0.4450
GFN	0.4436	0.4235	0.4333	0.4284
BUGE	**0.4578**	**0.4476**	**0.4526**	**0.4501**

Bold values represent the best experimental performance results

Table 5 Model efficiency comparison

Model	Memory usage (MiB)			Time consumption (S)		
	MR	IMDB	SST-5	MR	IMDB	SST-5
HyperGAT	523	1035	612	348	721	583
CGA2TC	629	1159	658	526	1232	741
BUGE	425	856	536	162	383	259

training process. However, the BUGE model we employ, by disregarding edge connections in the text graph, effectively addresses the problem of excessive smoothing in the model and can be efficiently extended to handle large-scale text sentiment corpora. During the sampling process, we also preserve both the global and local information within the graph structure adequately.

Analysis of efficiency

In this section, we investigate the efficiency of our model to validate its effectiveness. We compare the time and memory usage of different models, and the experimental results are presented in Table 5. We observed that in terms of time and memory usage, our model outperforms other models in comparison. This is because our model samples large heterogeneous graphs into smaller batches of edge-less subgraphs, removing the constraints imposed by edges, resulting in lower memory and time consumption.

Parameter sensitivity

In this section, we investigate the impact of different parameters on experimental results. While test recall and test accuracy serve as similar evaluation metrics for the model, we choose to showcase the model's performance using test accuracy on the dataset. We present the experimental results under various parameters on the dataset, and Fig. 2 illustrates the effect of different values of k on experimental accuracy. The value of k determines the size of the sampled neighbor nodes when performing subgraph sampling without edges, which can significantly affect the experimental outcomes.

We examine the results with different k values and observe that on the MR dataset, test accuracy gradually increases from $k = 1$ and then decreases after reaching the optimal value of $k = 16$. On the IMDB dataset, the optimal value is $k = 24$, and on the SST-5 dataset, the optimal value is $k = 16$. We have also noticed similar trends on other datasets. As the value of k increases, the efficiency cost of training the model gradually rises. However, in comparison to other GNN models, our model exhibits significantly reduced training costs.

Additionally, we also took into account the influence of other parameters on the experimental results. Figure 3 displays the accuracy of the MR and IMDB datasets at various window sizes. The graph illustrates that test accuracy initially rises as the window size increases. It reaches its peak when the window size is set to 20. However, once the window size surpasses 20, test accuracy starts to decline. Consequently, we have chosen to set the window size to 20.

We also considered the impact of vector dimensions on experimental results. We conducted comparative experiments using sentence vectors of varying dimensions on both the MR and IMDB datasets. Our objective was to analyze how the vector dimension affects the experimental outcomes. The results of these experiments are presented in Fig. 4.We can see from this that as the vector dimension increases up to 150 dimensions, the model's test accuracy reaches its maximum value. Beyond this point, the model's accuracy starts to gradually decrease. When the dimension of embeddings is too low, it may fail to effectively preserve the original features of the nodes. Conversely, embeddings with excessively high dimensions can demand more training time. Consequently, we have chosen to set the output dimension of the first layer to 150.

We also investigated the impact of different numbers of attention heads on the performance of neural networks, considering that multi-head attention is a crucial component in such models. To evaluate model performance, we used test accuracy as the metric for both the MR and IMDB datasets. The results, as presented in Fig. 5, indicate that our model achieved the highest test accuracy when utilizing six attention heads. Notably, the test accuracy reached its lowest point when the attention mechanism was not applied. Furthermore, the model's test accuracy gradually decreased as the number of attention heads exceeded 6. These experimental findings emphasize the importance of selecting an appropriate number of attention heads, as it can significantly improve the model's performance.

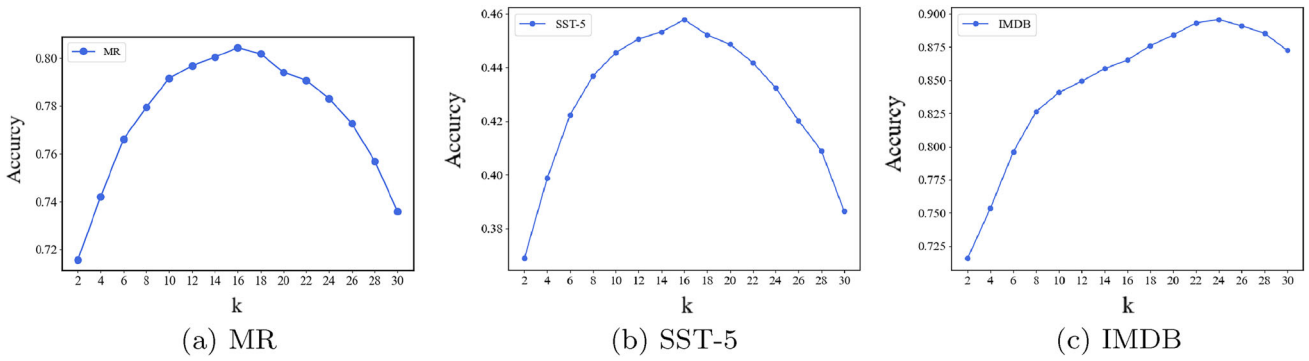

Fig. 2 The influence of k on the experimental accuracy under different values

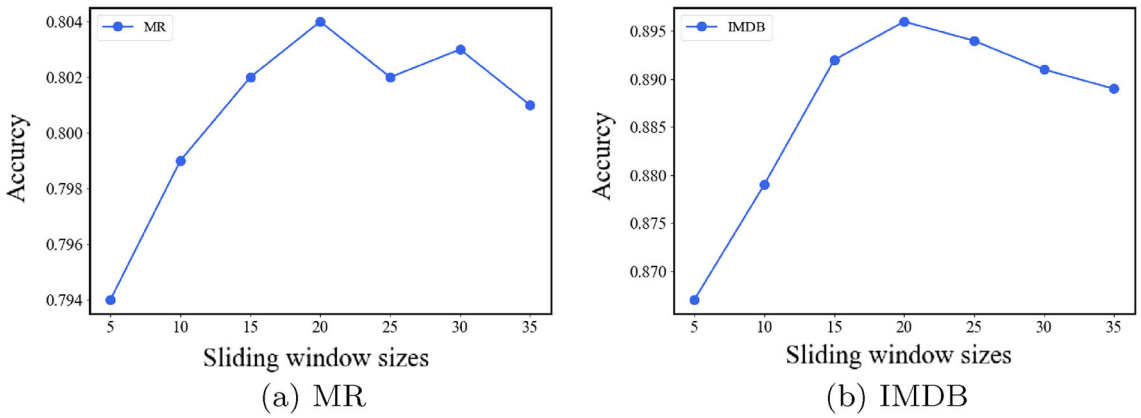

Fig. 3 The influence of sliding window sizes on the experimental accuracy under different values

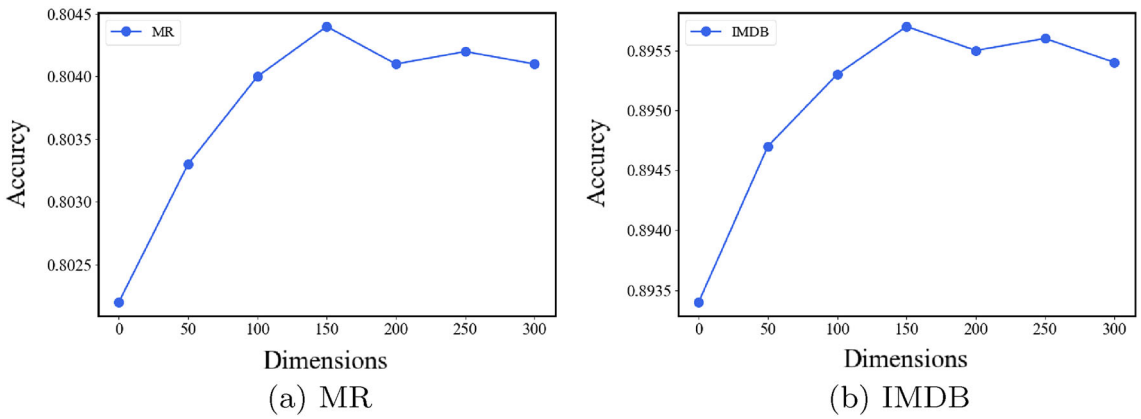

Fig. 4 The influence of dimensions on the experimental accuracy under different values

The learning rate is a crucial parameter in our experiment, and we explored different learning rate values to assess their effect on the model's performance. As depicted in Fig. 6, we can observe that the model achieves its best performance when the learning rate is set to 0.001. The experiment began with a learning rate of 0.1, and as the learning rate was reduced, the model's performance gradually improved. The optimal performance was achieved when the learning rate reached 0.001. Further reducing the learning rate led to a decline in the model's performance. Therefore, we have set the learning rate to 0.001.

Finally, we investigated the impact of the number of epochs on our experiments. The epoch number is a parameter that requires adjustment, and by conducting experiments with various numbers of epochs, our goal was to determine the optimal value for our specific task, which would help optimize model performance. According to Fig. 7, it can be observed that when the number of epochs is set to 50, the

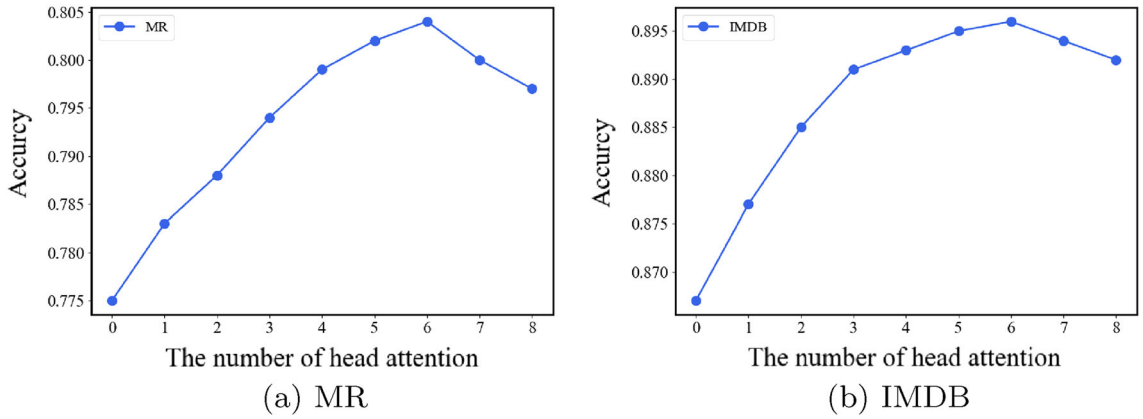

Fig. 5 The influence of head attention on the experimental accuracy under different values

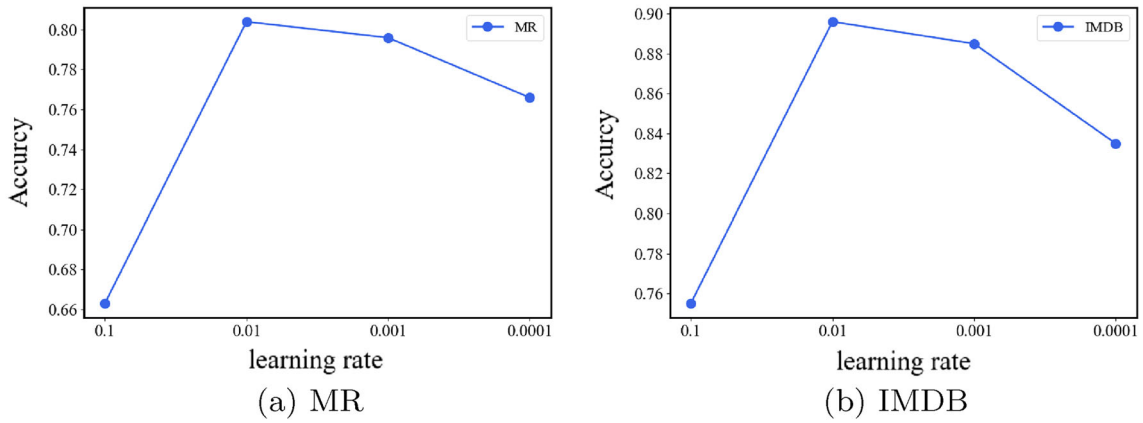

Fig. 6 The influence of learning rate on the experimental accuracy under different values

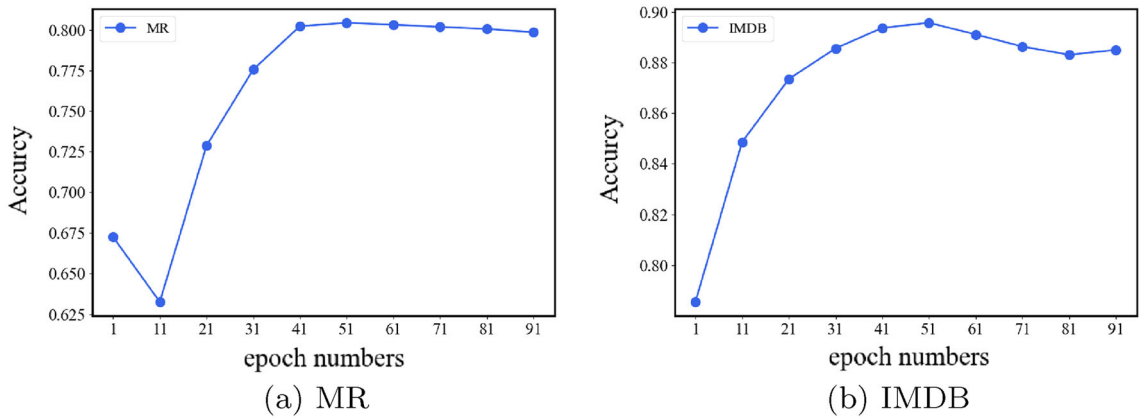

Fig. 7 The influence of epoch numbers on the experimental accuracy under different values

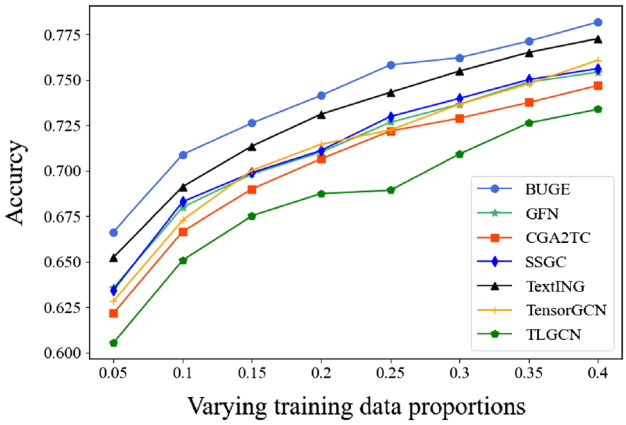

Fig. 8 The influence of varying training data proportions on the experimental accuracy under different values

model's performance reaches its peak. Further increasing the number of epochs does not significantly improve performance. This may be attributed to an excessive number of epochs, which can lead to overfitting and a reduction in the model's ability to generalize to new data.

Effects on the number of labelled data

To evaluate the influence of different proportions of training data on test accuracy, we conducted experiments using several models on MR datasets with varying proportions of training data. The comparison results are depicted in Fig. 8. Our BUGE model consistently demonstrated the best performance across all proportions of training data, achieving a test accuracy of 0.709 with only 10% of the training data. This underscores our model's capacity to perform well even with limited labeled data, showcasing its ability to effectively capture and retain textual information.

Conclusion and future work

In this paper, we introduce a novel Bert-based unlinked graph embedding (BUGE) model. Our approach demonstrates significant potential when handling large-scale text sentiment corpora graphs. By dividing the corpus into multiple unlinked subgraphs, each comprising a target node and its surrounding nodes without direct edge connections between them, our method enables representation learning that relies on attention mechanisms rather than graph connections. This effectively addresses the over-smoothing issue present in existing Graph Neural Network (GNN) models while enhancing model efficiency. Experimental results on multiple benchmark datasets have validated the effectiveness of our approach.

In the future, our research will focus on further improving the proposed BUGE model. We plan to integrate knowledge graphs into our graph pre-training model, enhancing interpretability by combining domain-specific knowledge graphs with data. This integration will enable better information retention and extraction when constructing text graphs, thereby further enhancing the model's performance.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant No. 62277040).

Data availability The data that support the findings of this study are openly available in Cornell University at https://www.cs.cornell.edu/people/pabo/movie-review-data/ and Stanford University at http://ai.stanford.edu/~amaas/data/sentiment.

Declarations

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this paper.

References

1. Saxena A, Reddy H, Saxena P (2022) Introduction to sentiment analysis covering basics, tools, evaluation metrics, challenges, and applications. In: Biswas A et al (eds) Principles of social networking: the new horizon and emerging challenges. Springer, Singapore, pp 249–277
2. Kaur R, Kautish S (2022) Multimodal sentiment analysis: a survey and comparison. Research anthology on implementing sentiment analysis across multiple disciplines. IGI Global, USA, pp 1846–1870
3. Yao L, Mao C, Luo Y. (2019) Graph convolutional networks for text classification. In: Proceedings of the AAAI conference on artificial intelligence, Honolulu, HI, USA, vol 33, pp 7370–7377
4. Niu L, Zheng Q, Zhang L (2021) Enhance gated graph neural network with syntactic for sentiment analysis. In: 2021 IEEE international conference on advances in electrical engineering and computer applications (AEECA), Dalian, China, pp 1055–1060
5. Niepert M, Ahmed M, Kutzkov K (2016) Learning convolutional neural networks for graphs. USA, PMLR, In: International conference on machine learning. New York, pp 2014–2023
6. Li Q, Han Z, Wu XM (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 32(1), New Orleans, Louisiana, USA

7. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN et al (2017) Attention is all you need. Adv Neural Inf Process Syst 30:5998–6008
8. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv arXiv:1609.02907
9. Huang L, Ma D, Li S, Zhang X, Wang H (2019) Text level graph neural network for text classification. arXiv arXiv:1910.02356
10. Ding K, Wang J, Li J, Li D, Liu H (2020) Be more with less: hypergraph attention networks for inductive text classification. arXiv arXiv:2011.00387
11. Zhu H, Koniusz P (2021) Simple spectral graph convolution. In: International conference on learning representations, Vienna, Austria
12. Zhang Y, Yu X, Cui Z, Wu S, Wen Z, Wang L (2020) Every document owns its structure: inductive text classification via graph neural networks. arXiv:2004.13826
13. Zhu X, Zhu L, Guo J, Liang S, Dietze S (2021) GL-GCN: global and local dependency guided graph convolutional networks for aspect-based sentiment classification. Expert Syst Appl 186:15712
14. Yang Y, Miao R, Wang Y, Wang X (2022) Contrastive graph convolutional networks with adaptive augmentation for text classification. Inf Process Manage 59(4):102946
15. Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv arXiv:1810.04805
16. Lu Z, Du P, Nie JY (2020) VGCN-BERT: augmenting BERT with graph embedding for text classification. Advances in information retrieval: 42nd European conference on IR research, ECIR 2020, Proceedings, Part I 42, Lisbon, Portugal, April 14–17
17. Yang Y, Cui X (2021) Bert-enhanced text graph neural network for classification. Entropy 23(11):1536
18. Lin Y, Meng Y, Sun X, Han Q, Kuang K, Li J, Wu F (2021) Bertgcn: transductive text classification by combining gcn and bert. arXiv arXiv:2105.05727
19. Shen H, Ju X, Chen X, Yang G (2023) EDP-BGCNN: effective defect prediction via BERT-based graph convolutional neural network. In: 2023 IEEE 47th annual computers, software, and applications conference (COMPSAC), Turin, Italy pp 850–859
20. Zhang J, Zhang H, Xia C, Sun L (2020) Graph-bert: only attention is needed for learning graph representations. arXiv arXiv:2001.05140
21. Zhang L, Meng Q (2019) Probabilistic ship domain with applications to ship collision risk assessment. Ocean Eng 186:106130
22. Pang B, Lee L (2005) Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. arXiv arXiv:cs/0506075
23. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, Florence, Italy, pp 1067–1077
24. Maas A, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, Portland, OR, USA, pp 142–150
25. Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng AY, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 conference on empirical methods in natural language processing, Seattle, WA, USA, pp 1631–1642
26. 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), Boston, MA, USA
27. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
28. Piao Y, Lee S, Lee D, Kim S (2022) Sparse structure learning via graph neural networks for inductive document classification. In: Proceedings of the AAAI conference on artificial intelligence, vol 36(10), Washington, USA, 7–14 February, pp 11165–11173
29. Liu X, You X, Zhang X, Wu J, Lv P (2020) Tensor graph convolutional networks for text classification. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, New York, NY, USA, pp 8409–8416
30. Dai Y, Shou L, Gong M, Xia X, Kang Z, Xu Z, Jiang D (2022) Graph fusion network for text classification. Knowl-Based Syst 236:107659