English Text Classification Model Based on Graph Neural Network Algorithm and Contrastive Learning

Chen Sian, Pan Guoqiang
Zhejiang Institute of Communications, Hangzhou, Zhejiang, 311112, China
E-mail: chen_sian82@outlook.com

Keywords: graph neural network, contrastive learning, english text classification, semantic representation, model construction

Recieved: March 1, 2025

Current English text classification methods mostly rely on bag-of-words models or CNN (Convolutional Neural Network), but there are limitations in processing text structure and semantics. Especially in long texts and complex contexts, it is difficult to capture the long-distance dependency and structured semantics between words. To this end, this article combines GNN (Graph Neural Network) with contrastive learning to build an English text classification model. First, a text graph is constructed through word co-occurrence to capture the long-distance dependency of words. Then, a multi-layer graph convolutional network is designed, and residual connections and normalization are applied to improve model performance. A contrast learning module is added after each layer of graph convolution to improve node features and semantic representation. Triplet Loss is a loss function, and Hard Negative Mining chooses negative samples to improve efficiency.

Povzetek: Predlagan je model za klasifikacijo angleškega besedila, ki združuje grafične nevronske mreže (GNN) in kontrastno učenje (CL). GNN-ji s pomočjo ko-pojavitvene matrike ustvarijo graf za zajemanje medsebojnih odvisnosti besed. CL (z izgubo Triplet Loss) izboljša semantično reprezentacijo vozlišč GNN, kar model (CS-K-prototipi) pri klasifikaciji besedil bistveno izboljša natančnost in robustnost.

1 Introduction

English text classification is critical for natural language processing, affecting many aspects such as information retrieval, sentiment analysis, and question-answering systems. The importance of text classification is increasing with the proliferation of networks and information content. Although traditional methods such as bag-of-words models and TF-IDF (term frequency-inverse document frequency) have some effects, they cannot cope with deep semantics, long texts, or complex contexts. Although deep learning methods such as CNN and RNN (Recurrent Neural Network) have progressed, they still cannot perfectly handle long-distance dependencies. Graph Neural Networks (GNN) and contrastive learning are two deep learning-based pattern recognition algorithms that improve accuracy, adaptability, and efficiency over rule-based approaches in software testing. Complex data linkages are captured, unstructured data is handled, manual feature engineering is decreased, and generalization is improved. To enhance text classification, new methods and technologies have emerged. GNN can capture structured information, simulate complex relationships between words, and deepen text understanding. Contrastive learning is a strategy for improving model performance by comparing data samples and moving similar ones closer together while pushing dissimilar ones apart. In this study, semantic embeddings are optimized using Triplet Loss, which improves text categorization accuracy. Contrastive learning, a self-supervised strategy, can strengthen the feature representation of the model and reduce the dependence on labeled data. This study combines GNN and contrastive learning to create a new English text classification model. Graph Neural Networks have improved at dealing with extensive texts and intricate interactions than models such as CNNs and RNNs because they can represent text as a graph, with words as nodes and relationships as edges. This enables GNNs to capture long-distance dependencies and intricate semantic linkages between words, which CNNs struggle with due to their reliance on fixed-size filters, while RNNs suffer with long sequences due to concerns such as vanishing gradients. GNNs improve their understanding of a text's global structure and deep semantics by pooling input from surrounding nodes. Combining GNNs with contrastive learning improves feature representation, allowing for accurate and robust handling of complicated and lengthy texts. This model can not only effectively grasp the grammatical and semantic relationships of the text but also improve semantic embedding through contrastive learning, thereby improving classification accuracy. This study has injected new vitality into the field of text classification and promoted the advancement of related technologies. Semantic embedding transforms text into a vector space in which comparable words are closer together, allowing the model to better grasp word associations. By integrating Graph Neural Networks (GNN) and contrastive learning, the study improves semantic embeddings, allowing the model to capture

complex relationships and long-distance dependencies for better text classification.

Text semantic extraction helps precisely identify the text's relationship and structure and provides intelligent support for information retrieval, sentiment analysis, etc. Improving semantic understanding can improve the accuracy of natural language processing tasks and meet personalized information needs [1-2]. To solve the problem of text semantic information extraction, many scholars have proposed different methods. Martinez-Rodriguez J. L proposed a strategy for extracting information from sentences and representing it using semantic network standards. The strategy involved information extraction tasks and hybrid semantic similarity metrics. Experiments proved the proposed method's feasibility and accuracy [3]. Current Chinese short text entity linking techniques ignore the interaction between label information and the original short text and fail to effectively utilize semantic information. Gao L proposed a normalization method to fully extract semantic information from short text sentence vectors. The results showed that the proposed model outperformed popular deep-learning techniques and previous research results in entity linking [4]. The process of automatically determining the connection between two or more elements is called semantic relation extraction, which is crucial for creating original writing. In addition to describing established and new evaluation metrics for supervised, semi-supervised, and unsupervised methods, Gharagozlou H also studied several relation extraction techniques and types in English and the most popular techniques in Persian [5]. Accurate identification and analysis of semantics are conducive to effectively processing English text. Yu S introduced Word2vec (word to vector) for extracting semantic feature vectors from English text and the long short-term memory (LSTM) algorithm for semantic identification of English text. The results showed that the identification results of the LSTM algorithm for the part of speech and sentiment tendency of English text were consistent with the label results [6]. Scholars emphasize semantic information and use different techniques to improve model performance. However, the research has not fully utilized the graph structure to capture the complex relationship of text. Sequential models, which concentrate mostly on local aspects and may have trouble with long-distance dependencies, frequently miss deep semantic linkages that graph structures capture. A graph structure preserves both local and global dependencies by representing words as nodes and their interactions as edges, in contrast to CNNs and RNNs that analyze text sequentially. This makes it possible to comprehend text semantics more precisely, especially in intricate circumstances. This representation is further improved by combining contrastive learning with Graph Neural Networks (GNNs), which increases the model's capacity to differentiate between text categories and boosts performance on tasks with intricate semantics and long-distance dependencies. It has limitations in the processing of inter-lexical dependencies and deep semantic structures.

GNN can better capture the relationship and meaning between words by turning text into a graph, enhance the model's understanding of text structure, and improve classification accuracy, especially in processing long articles and tasks that require an understanding of context [7-8]. In recent years, some researchers have used GNN in text classification research. To solve the problem of cross-lingual text classification, Vo T proposed a new topic-driven multi-type text graph attention representation learning technology, which combined neural topic modelling technology with a heterogeneous text graph attention network to enhance the semantic information of text representation learned in various language environments. GAT and GraphSAGE are two models with distinct advantages in text classification problems. GAT incorporates an attention mechanism into graph convolutional layers, allowing the model to focus on meaningful words or relationships, hence enhancing accuracy. GraphSAGE minimizes computational complexity by sampling neighbors during training and enhances scalability, particularly for large-scale graphs. Its aggregation approaches, such as mean, pooling, and LSTM-based aggregators, enable the model to capture broad semantic patterns while avoiding overfitting. When coupled, these models could offer a more robust method for dealing with complicated semantics and long-distance interdependence.

The proposed model was compared with the current state-of-the-art baseline and experimentally demonstrated its effectiveness [9]. Deng Z proposed a new graph-based model and designed an attention-gated graph neural network to propagate and update the semantic information of each word node to solve the problem that existing methods are not enough to capture the semantic relationship between words. Experimental results showed that the proposed model outperformed previous text classification methods [10]. Parthasarathy (2023) examines combining neural networks with the Harmony Search Algorithm (HSA) to improve fraud detection in banking. Traditional methods often fail against complex fraud techniques, but this combination enhances accuracy and reliability. The findings suggest that models like Decision Tree Classifier and Sequential models, with near-perfect accuracy, could transform fraud prevention. However, this study supports the idea that combining neural networks with the Harmony Search Algorithm (HSA) to improve fraud detection parallels the approach in our work to enhance accuracy and reliability in text classification [11]. The above scholars have cleverly used

graph structures and attention mechanisms to grasp the semantic relationship of text, significantly improving the model's ability to handle complex semantics and performing well in cross-language text classification. However, the graph neural network's capture of deep semantics and long-distance dependencies needs to be strengthened, and it has not fully utilized contrastive learning to enhance feature representation.

This study combines GNN with contrastive learning to deeply capture word relationships by constructing a text graph and optimizing semantic embedding using contrastive learning. GNN converts text into a graph with words as nodes and edges showing semantic connections. The results show that this method can deeply capture text semantics and long-distance dependencies, significantly improve performance in multiple text classification tasks, and demonstrate its excellent generalization ability and robustness. Compared with CNN, this method is more precise and stable when dealing with complex text classification. The innovation of this study is to combine GNN with contrastive learning for English text classification, which makes up for the shortcomings of traditional methods and reduces the dependence on labeled data. The new graph contrast loss function captures text semantics more precisely. The graph contrast loss function has numerous significant advantages over ordinary contrastive loss functions, especially in the context of graph neural networks (GNN) and contrastive learning for text categorization. It takes advantage of the network structure of text to improve the model's capacity to capture semantic linkages between words, addressing the complex, long-distance dependencies that typical contrastive loss functions frequently overlook. By taking into account both pairwise similarities and contextual relationships within the network, the graph contrast loss function improves semantic embedding quality, resulting in more accurate and informative text representations. The function also includes Hard Negative Mining, which concentrates on difficult-to-detect negative samples, allowing the model to acquire more discriminatory features and enhance generalization. At the same time, through strategy optimization and parameter adjustment, the classification accuracy and training efficiency are improved. These innovations have promoted the development of text classification technology and provided new ideas for natural language processing. The organizational structure of this study is shown in Figure 1:

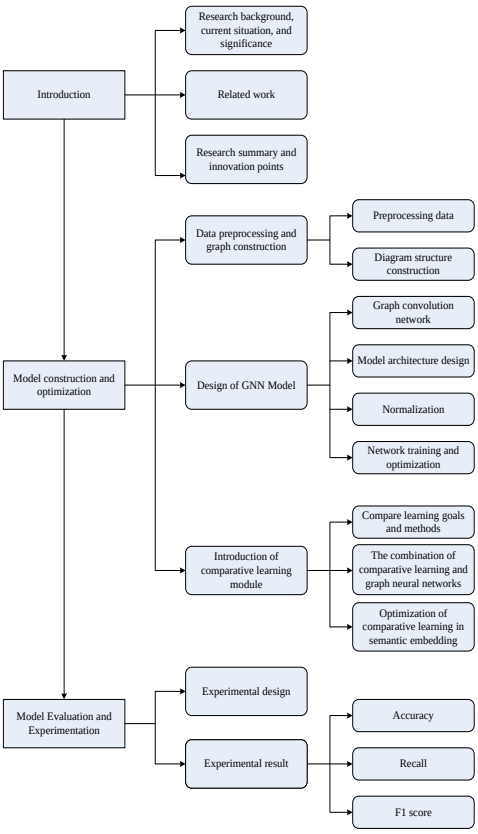

Figure 1: Organizational structure of this study

2 Construction and optimization of the english text classification model

2.1 Data Preprocessing and graph construction

2.1.1 Preprocessing data

Data preprocessing is critical for English text classification. It can clean text, remove noise, and convert it into a format suitable for GNN. The proposed approach combines contrastive learning with Graph Neural Networks (GNN) to handle noisy or redundant test instances efficiently. GNN focuses on pertinent semantic connections while capturing long-distance dependencies and deep semantic interconnections between words. By differentiating between comparable and dissimilar samples, contrastive learning improves the resilience of the model. By focusing on hard-to-classify negative samples, hard negative mining improves classification accuracy and lessens the influence of redundant data.

This article cleans, segments, removes stop words, and stems the data to extract valuable semantic information, laying the foundation for subsequent graph construction and graph neural network training. Text cleaning is the first step in data preprocessing. The original text contains many irrelevant information, such as punctuation, numbers, etc., which may interfere with the analysis. This article uses regular expressions to clean up these noises, retaining only letters and spaces to ensure the text's purity. The next step is segmenting the text into independent words or phrases. Using the word tokenize method of NLTK (Natural Language Toolkit), word segmentation is performed by space and punctuation, and the text is converted into a vocabulary list. Afterword segmentation, stop words are filtered out. Removing stop words can reduce the amount of calculation and prevent the model from being interfered with by irrelevant information. NLTK stop word library is utilized for filtering. After that, stemming is done to normalize different forms of words to the basic form, such as "running" becomes "run". This can reduce the number of words and vocabulary dimensions and improve training efficiency. Then, the Porter stemming algorithm and the Porter Stemmer class of NLTK are processed. After this preprocessing, the original text becomes a preprocessed standardized vocabulary list.

Word segmentation, stop word elimination, and stemming are critical processes that ensure efficient processing of data in order to prepare it for text classification model operation. Since the text is represented as a graph with words as nodes in models like Graph Neural Networks (GNNs), word segmentation is crucial since it separates the text into discrete words or tokens. By getting rid of popular but useless words, stop word removal lowers computing costs and directs the model's attention to more important data. Stemming minimizes vocabulary quantity and increases training efficiency by breaking words down to their most basic forms.

2.1.2 Graph structure construction

Each document is treated as a graph. Among them, words correspond to nodes; edges between nodes represent semantic relationships between words; edge weights represent the strength of the relationship. After data preparation, a text graph structure is constructed for GNN. In this study, edges are built based on word co-occurrence information, and the co-occurrence matrix is used to quantify the word association. Semantic granularity and computational performance must be balanced when choosing the window size for co-occurrence computation in the word co-occurrence network. For tasks like text categorization, a larger window size aids in capturing broader, long-distance semantic dependencies, whereas a smaller window size captures local, syntactic interactions between close words. Depending on the needs of the text, the window size is selected; larger windows make it easier to record intricate relationships in lengthy texts. The sliding window size v is first set to construct the co-occurrence matrix, determining which words are closely related semantically. Words that co-occur within a window are considered related. If two words appear in the same window, they are semantically related. The co-occurrence matrix D is symmetric, and the element d_{ij} represents the number of times words v_i and v_j co-occur in the window. Each document window is traversed, and the number of co-occurrences of each pair of words is calculated to construct a matrix. The formula (1) is:

$$d_{ij} = \sum_{l=1}^{M-v+1} \vartheta\left(\mathrm{v}_i, l\right) \cdot \vartheta\left(\mathrm{v}_j, l+1\right) \quad (1)$$

Among them: $\vartheta(\mathrm{v}_i, l)$ and $\vartheta\left(\mathrm{v}_j, l+1\right)$-the indicator functions;

The total number of words in the document;

v-the set sliding window size.

If the words v_i and v_j appear adjacent in the text (v_i is at position l, and v_j is at position l+1), the function value is marked as 1. Otherwise, it is marked as 0. Based on this method, a symmetric matrix can be constructed, whose element d_{ij} represents the co-occurrence frequency of v_i and v_j in a given window, reflecting the closeness of their semantic connection.

When generating the graph structure, the words in the document are represented as nodes, and the co-occurrence matrix determines the edge weights to capture the long-distance dependencies between words. Unlike the traditional bag-of-words model, the graph structure retains the order of words and effectively displays complex semantic connections, providing rich information for graph neural networks. PMI (Pointwise Mutual Information) is used to measure the similarity of word pairs to enhance the graph structure [12-13]. The formula (2) for PMI is:

$$\mathrm{PMI}\left(\mathrm{v}_i, \mathrm{v}_j\right) = \log \frac{Q(\mathrm{v}_i, \mathrm{v}_j)}{Q(\mathrm{v}_i) Q(\mathrm{v}_j)} \quad (2)$$

Among them: $Q\left(\mathrm{v}_i, \mathrm{v}_j\right)$-the joint probability of words v_i and v_j appearing in the document at the same time;

$Q(\mathrm{v}_i)$ and $Q\left(\mathrm{v}_j\right)$-the marginal probabilities of words v_i and v_j.

The joint probability $Q\left(\mathrm{v}_i, \mathrm{v}_j\right)$ is derived from the elements of the co-occurrence matrix, and the marginal

probabilities $Q(\mathrm{v}_i)$ and $Q(\mathrm{v}_j)$ are estimated based on the occurrence frequency of the words in the document.

Semantically related word pairs can be identified by calculating PMI, and corresponding edges can be established in the graph. The two words can be connected only when the PMI value exceeds the set threshold. A text graph is created using words as nodes and edges signifying semantic associations in order to weight word correlations and perform edge pruning. Pointwise Mutual Information (PMI), which gauges how similar word pairings are to one another, and a co-occurrence matrix are used to assess how strong these links are. Stronger semantic connections are captured when words with a PMI value above a threshold are joined by edges. By eliminating shoddy or irrelevant connections, edge pruning improves the graph's performance and the quality of the semantic embedding. In this way, the text graph structure can precisely model the deep relationship between words and provide accurate and rich data to the graph neural network, thereby improving the performance of classification tasks. The formation of text preprocessing to graph structure is shown in Figure 2:

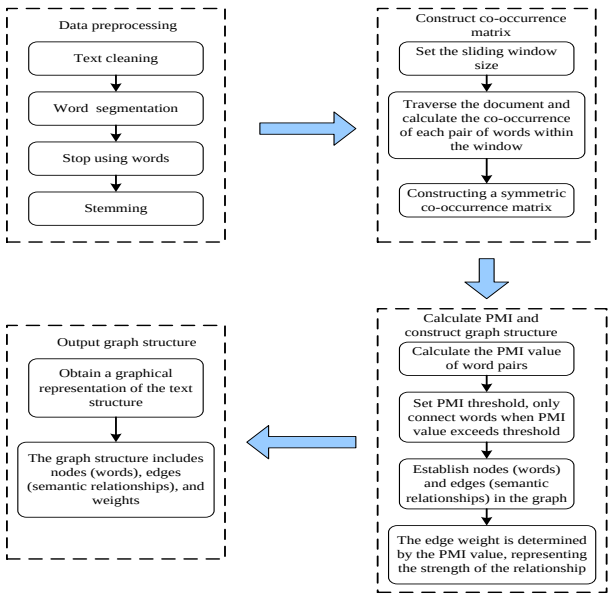

Figure 2: The process of forming a graph structure from text preprocessing

2.2 Graph neural network model design

2.2.1 Graph convolutional network

When designing a GNN model, first, a graph structure is built based on the text, and then, GCN is used to propagate information and learn features, aiming to deeply capture the semantics and long-distance dependencies of the text. A multi-layer GCN architecture is adopted to enhance the model's performance, and residual connections and normalization strategies are added to ensure information flow and prevent gradient disappearance. GCN is an effective method for processing graph-structured data and performs well in graph-related tasks [14-15]. Graph neural networks (GNNs) benefit significantly from residual connections, especially when it comes to solving the problem of gradient vanishing in deep designs. The gradients don't decrease during backpropagation, which is a common problem in deep networks, because to these links, which allow information to travel directly between layers. In the absence of residual connections, deeper models have trouble with gradient propagation, which can lead to poor convergence or unsuccessful training. Residual connections provide more effective feature learning and preserve stable training by letting gradients avoid layers. They play a crucial role in GCN-based models by maintaining pertinent data across layers, which enhances the network's capacity to represent intricate and distant connections in text. In the English text classification task, GCN effectively captures the text's deep semantics and long-distance dependencies through the graph structure. Words are regarded as nodes, and edges represent the relationship between words. The graph convolution operation of GCN enables the model to propagate and learn node information and then deeply understand the semantics of words in context.

If the features of the nodes in the graph are represented by $g_i^{(k)}$, representing the features at the k-th layer, GCN updates them according to Formula (3).

$$g_i^{(k+1)} = \delta\left(\sum_{j\in N(i)} \frac{1}{|N(i)|}\cdot\frac{1}{|N(j)|} U^{(k)} g_j^{(k)} + U_0^{(k)} g_i^{(k)}\right) \quad (3)$$

Among them: $N(i)$-the set of neighbor nodes of the node;

$U^{(k)}$ and $U_0^{(k)}$-the learnable parameters of the k-th layer;

δ-the nonlinear activation function is ReLU (Rectified Linear Unit).

$g_i^{(k)}$-represents the features of node i at layer k.

$\frac{1}{|N(i)|}$- normalizes the aggregation of neighbor features.

2.2.2 Model architecture design

When designing the GCN model, multiple layers of GCN are stacked to enhance the expression ability and feature depth. Each layer updates the node features by aggregating neighbor information, capturing complex relationships more deeply than a single layer. The multi-layer Graph Convolutional Network (GCN) is designed with the primary goal of efficiently capturing long-distance connections and semantic linkages in text. To capture both local and global semantic patterns, the model employs a multi-layer GCN architecture, in which each

layer collects data from nearby words (nodes). In order to ensure efficient information transfer between the layers and prevent gradient vanishing, residual connections are included. By preserving constant feature scales, normalization approaches are used to stabilize training and enhance convergence. The graph-based structure improves feature representation by enabling information to spread through semantic relationships between words. At each layer, contrastive learning is also used to further improve semantic understanding by differentiating between similar and dissimilar text categories using a Triplet Loss function.

The performance of the Graph Convolutional Network (GCN) model in text categorization is greatly improved by its depth, which includes many layers, residual connections, and normalization. The model's several layers enable it to capture intricate, far-reaching semantic relationships between words. However, residual connections ensure that the gradient flow is maintained during backpropagation, which helps to avoid the vanishing gradients that might affect deeper networks. By guaranteeing uniform feature distributions among layers, normalization enhances convergence stability and speed, further stabilizing training.

To solve the problem of gradient disappearance caused by multiple layers, residual connections are applied to ensure effective information transmission. The graph convolution update formula (4) is:

$$g_i^{(k+1)} = g_i^{(k)} + \delta\left(\sum_{j\in N(i)} \frac{1}{|N(i)|}\cdot\frac{1}{|N(j)|} U^{(k)} g_j^{(k)} + U_0^{(k)} g_i^{(k)}\right) \quad (4)$$

By stacking multiple layers of GCN, the model can learn richer node representations and integrate local and global information. After GCN processing, word feature representations can more precisely capture the complex semantics of the text and help text classification. This architecture improves model performance, effectively copes with complex semantics in large-scale text data, and achieves efficient processing. The model structure of this article is shown in Figure 3.

ResNet is a deep residual network architecture that enhances automated test case generation by improving accuracy and efficiency. It overcomes challenges like vanishing gradients, allowing deeper networks to train without losing important information. ResNet captures hierarchical features and retains essential data through residual connections, making it useful for complex data structures. It generates diverse test cases, including edge cases, and ensures each network layer contributes to better feature extraction, resulting in more accurate, reliable, and efficient test case generation for robust software validation.

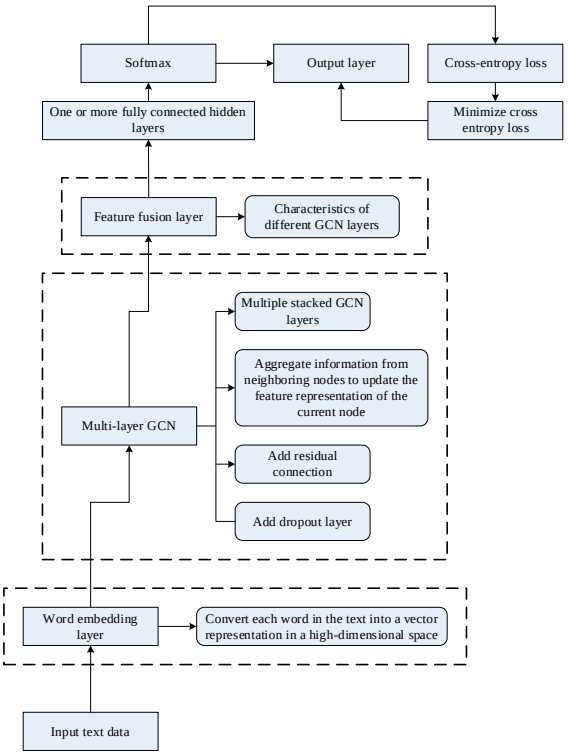

Figure 3: Model structure diagram

The proposed approach addresses the vanishing gradient issue, increases training stability, and speeds up convergence by utilizing ResNet layers to enhance pattern recognition. While deeper layers concentrate on intricate patterns like object pieces or semantic structures, early layers capture basic aspects like edges and textures. Even in deep networks, residual connections allow for the effective learning of both low-level and high-level information, leading to more reliable and accurate text classification.

2.2.3 Normalization in graph convolutional networks

Normalization operations are added to the model to improve the training stability and convergence speed of GCN. The heterogeneity between nodes in graph structure data leads to large differences in node feature distribution, affecting training efficiency and performance. Therefore, layer normalization technology ensures that the scale of input features of each convolution layer is similar. Layer normalization independently normalizes the features of each layer of nodes to ensure that the input features are evenly distributed and have consistent scales. Contrastive learning refines Graph Convolutional Networks (GCNs) to improve text categorization performance by optimizing the text's semantic representations.

GCNs capture semantic relationships by converting text to a graph structure while retaining long-distance interdependence. Contrastive learning, using Triplet Loss, refines these embeddings by bringing comparable text samples closer together and pushing dissimilar ones apart, hence boosting classification accuracy. Furthermore, Hard Negative Mining concentrates on difficult-to-distinguish negative data, speeding up the learning process and improving the model's capacity to detect minor semantic differences. The temperature parameter helps to stabilize training by regulating gradient updates, resulting in smoother learning and preventing abrupt changes early on.

Unlike batch normalization, layer normalization does not rely on batch statistical information and is more suitable for graph data. Neural network training is stabilized and accelerated by the use of Layer Normalization (LN) and Batch Normalization (BN). Because LN normalizes inputs across properties of each individual data point, it can be used with graph-based models. It guarantees that the feature representation of every node is stable and performs well with tiny or irregular batches. BN uses batch statistics to normalize the entire batch, which may not be as successful because of differences in node properties and graph sizes. In graph-based models, LN is favored because it individually normalizes the properties of each node, resulting in more stable and efficient training, particularly in graph data that is sparse and volatile.

The layer normalization formula (5) is:

$$\hat{g}_i^{(k)} = \frac{g_i^{(k)} - \varphi^{(k)}}{\delta^{(k)}} \cdot \alpha^{(k)} + \beta^{(k)} \quad (5)$$

Among them: $\varphi^{(k)}$ and $\delta^{(k)}$-the mean and standard deviation of the features of the k-th layer;

$\alpha^{(k)}$ and $\beta^{(k)}$-the learnable scaling and offset parameters;

$\hat{g}_i^{(k)}$-the normalized features.

Dropout is added after each layer to improve the generalization ability by randomly discarding some connections to prevent GCN from overfitting. With the normalization operation, GCN is more robust when processing high-dimensional data and complex tasks, improving the generalization performance and training efficiency of English text classification and ensuring that the model is stable and has strong generalization ability.

2.2.4 Training and optimization of graph convolutional networks

During the training process of GCN, supervised learning is used, and the classification effect is optimized by minimizing the cross-entropy loss. This loss function can measure the gap between the predicted result and the true label. In the English text classification task, word features are regarded as graph nodes and information propagation and update are realized through GCN. The loss function formula (6) is expressed as:

$$K = -\sum_{i=1}^{M} \sum_{d=1}^{D} b_{i,d} \log\left(\hat{b}_{i,d}\right) \quad (6)$$

Among them: M-the number of samples;

The number of categories;

$b_{i,d}$-the true label of the i-th sample in category d;

$\hat{b}_{i,}$ the prediction probability of the model in category d.

To improve the training speed and optimization effect, the Adam (Adaptive Moment Estimation) optimizer is selected, which can dynamically adjust the learning rate according to the mean and variance of the gradient, thereby achieving faster convergence and preventing gradient problems. The updated rules are shown in Formulas (7) to (10):

$$n_r = \gamma_1 n_{r-1} + (1 - \gamma_1) h_r \quad (7)$$

$$w_r = \gamma_2 w_{r-1} + (1 - \gamma_2) h_r^2 \quad (8)$$

$$\hat{n}_r = \frac{n_r}{1 - \gamma_1^r}, \hat{w}_r = \frac{w_r}{1 - \gamma_2^r} \quad (9)$$

$$\eta_r = \eta_{r-1} - \mu \frac{\hat{n}_r}{\sqrt{\hat{w}_r + \epsilon}} \quad (10)$$

Among them: n_r and w_r-the mean and variance of gradient;

h_r-the gradient at the current moment;

γ_1 and γ_2-the hyperparameters, used to control the decay rate of the first-order moment estimate and the second-order moment estimate;

μ-the learning rate;

ϵ-the small constant to prevent zero division errors.

Using the Adam optimizer, GCN can adaptively adjust the parameter update step to avoid the limitations of traditional gradient descent, such as learning rate sensitivity and gradient explosion, thereby improving convergence speed and model stability.

2.3 Application of contrastive learning module

2.3.1 Objectives and methods of contrastive learning

To improve the performance of GNN in text classification, this study applies a contrastive learning mechanism. This mechanism optimizes semantic representation by maximizing the distance between texts of different categories, making texts of the same category closer and texts of different categories more distant, which helps GNN capture long-distance dependencies and

complex semantics. The proposed discusses how Graph Neural Networks (GNNs) form node representations by using semantic relationships between words, transforming text into a graph where words are nodes and edges represent their semantic connections. GNNs capture long-distance dependencies and complex relationships through graph convolution operations. Additionally, contrastive learning enhances these representations by refining the similarity between words of the same category and distinguishing those from different categories.

Contrastive learning performs an important role in decreasing noise and improving the quality of semantic embeddings in text classification because it optimises semantic representations by enhancing the distance between different samples and minimizing the distance between comparable ones. This strategy enhances the model's capacity to identify between categories, particularly when dealing with noisy or ambiguous input. In this study, contrastive learning, in conjunction with Graph Neural Networks (GNN), refines semantic features and aids in the capturing of deep word associations. Hard Negative Mining prioritizes difficult negative samples, enhancing the model's learning efficiency, and temperature parameters stabilize training by managing gradient updates. Contrastive learning does not rely on traditional annotations and provides greater flexibility and adaptability. This study uses Triplet Loss as the loss function, which aims to reduce the distance between the anchor point and the positive sample and increase the distance between the anchor point and the negative sample, significantly improving the accuracy and efficiency of GNN in text classification. The formula (11) is:

$$L_{triplet} = max(e(x,p) - e(x,n) + \lambda, 0) \quad (11)$$

Among them: λ-the feature representation of anchor samples;

p and the feature representations of positive samples and negative samples.

By minimizing Triplet Loss, the model can make similar samples closer and heterogeneous samples more distant in the embedding space, thereby improving the accuracy of text classification. Samples with similar or the same labels are selected as positive samples, and samples with different or low similarity are selected as negative samples. The Comparing loss functions like Triplet Loss and NT-Xent is essential for evaluating classification performance. The model's capacity to differentiate between classes is improved by both loss functions, which modify the separation between sample representations. Because of its ease of use and function in keeping anchor samples far from negative ones and closer to positive ones, triplet loss is prized. However, NT-Xent Loss is more successful at identifying minute variations across classes because it adds a temperature parameter that gives it more accurate control over the embedding space. Although Triplet Loss was chosen for this study because of its efficacy, a comparison with NT-Xent may provide more information on how each model contributes to performance, especially in cases with complicated semantics and long-distance dependencies. Optimizing this loss function helps the model learn more precise text representation.

2.3.2 Combination of contrastive learning and graph neural network

In GNN, text is converted into a graph structure, with vocabulary represented by nodes and relationships represented by edges. GCN learns node features, and contrastive learning optimizes semantic dependencies. Node characteristics are improved for text classification using contrastive learning in a GNN by transforming text into a graph with nodes representing words. A multi-layer GCN captures semantic dependencies, but contrastive learning using Triplet Loss reduces the distance between similar phrases while increasing it for different ones in the embedding space. Hard Negative Mining concentrates on tough negative data, and a temperature parameter smoothes the loss function for more stable training. This combination enables the model to capture the comprehensive semantics of the text. Contrastive learning is added after each layer of graph convolution to improve the discriminability of text representation. Node features are regarded as global feature training, and similarity is calculated based on node embedding so that GNN can extract local and overall semantics at the same time. During training, the model optimizes the graph structure and node features through contrastive learning to capture semantics more precisely.

Furthermore, combining contrastive learning and GNN, a new graph contrast loss function is designed to consider node similarity and category information to improve the accuracy of semantic understanding. The model extracts feature with GCN and then optimizes with this function to enhance text classification performance. The optimization formula (12) of the graph contrast loss function is:

$$L_{triplet} = \sum_{i=1}^{M} \sum_{j=1}^{M} \left[e\left(f_i, f_j\right) - \lambda \cdot I\left(b_i \neq b_j\right) \right] \quad (12)$$

Among them: f_i and f_j-the node feature representation;

b_i and b_j-the node category labels;

$e(f_i,\ \ )$ the distance measurement between nodes.

2.3.3 Optimization of contrastive learning in semantic embedding

This study integrates category information into contrastive learning to improve the quality of semantic embedding, making similar texts closer and different categories more separated. Contrastive learning refines Graph Convolutional Networks (GCNs) to improve text categorization performance by optimizing the text's semantic representations. GCNs capture semantic relationships by converting text to a graph structure while retaining long-distance interdependence. Contrastive learning, using Triplet Loss, refines these embeddings by bringing comparable text samples closer together and pushing dissimilar ones apart, hence boosting classification accuracy. Furthermore, Hard Negative Mining concentrates on difficult-to-distinguish negative data, speeding up the learning process and improving the model's capacity to detect minor semantic differences. The temperature parameter helps to stabilize training by regulating gradient updates, resulting in smoother learning and preventing abrupt changes early on. The Hard Negative Mining strategy is adopted to focus on negative samples that are difficult to distinguish. Unlike traditional methods, this strategy selects negative samples based on model performance to improve learning efficiency. The dynamic selection of 'hard' negative samples in contrastive learning concentrates on the most difficult cases that are closest to the anchor sample. This method, known as Hard Negative Mining, increases model discriminative power, speeds up training convergence, and improves generalization. It helps the model better discern insignificant distinctions, especially in complex or imbalanced datasets, resulting in more efficient and robust performance in tasks like as text categorization. Hard Negative Mining, which concentrates on choosing negative examples that are challenging to distinguish, improves the negative sample selection procedure in this research. By pushing the model to learn from difficult examples rather than simple negatives, this technique increases the discriminability of the model and produces more robust and instructive representations. By lowering the possibility of overfitting to readily classifiable negative samples, it also helps to maintain the stability of the model. Furthermore, the contrastive loss function's incorporation of a temperature parameter regulates the gradient updates' smoothness, avoiding drastic changes early in the training process and encouraging steadier optimization. By focusing on such samples, the model can better capture the subtle differences in text features and enhance classification capabilities.

Hard Negative Mining (HMN) is a technique used to enhance model learning by choosing the most difficult negative samples—those that are hard to differentiate from positive ones. HMN highlights the most instructive negative examples, in contrast to random negative mining, which chooses negative samples independent of their proximity to the decision boundary, or semi-hard negative mining, which targets samples near but not on the boundary. By making the model pick up on minute differences, this method speeds up model convergence and decreases overfitting. HMN improves the contrastive learning framework and Graph Neural Network (GNN) semantic embedding in the study, increasing classification robustness and accuracy, especially for challenging tasks like text categorization.

In each round of training, Hard Negative Mining optimizes the negative samples closest to the anchor point. Hard Negative Mining focuses on negative samples that are hard to separate from positive samples by choosing those that are closest to the anchor point in feature space. Metrics like cosine similarity or Euclidean distance are frequently used to measure the distance or similarity between the feature vectors of the samples. By choosing these difficult negative samples, the model improves its generalization skills by learning to distinguish between classes more precisely. This strategy encourages the model to focus on those difficult-to-distinguish samples in the feature space, thereby performing more precise feature identification and improving the accuracy of text classification. In addition, it prevents the model from paying too much attention to samples that are easy to classify, thereby reducing the risk of overfitting. Therefore, the application of difficult negative samples not only does it improve the model's classification ability and accelerate training convergence, but its advantages become more evident when processing complex texts. This article applies a temperature parameter to optimize the contrastive learning process. The intensity of gradient updates is controlled by smoothing the loss function to maintain training stability. The temperature parameter adjusts the influence of the distance between samples, enhances the robustness of the loss function, and avoids extreme gradient updates in the early stage of learning. The temperature loss function formula (13) is:

$$L_{contrastive} = \frac{1}{T}\log\left(1 + exp\left(\frac{e(x,n)}{T}\right)\right) \quad (13)$$

Among them is T- the temperature parameter, which controls the smoothness of the loss function.

By changing the temperature parameters, the model can optimize the contrastive learning effect, prevent it from entering local optima, and adjust the learning speed and gradient changes according to the training stage and sample difficulty. The model's optimal temperature parameter was demonstrated to improve contrastive learning's semantic embedding quality and training stability. It ensures smoother convergence by preventing problems like excessive gradients in the early phases of training by regulating the degree of gradient updates. By

controlling the distance between samples, the temperature balances the impact of both simple and complex examples. The robustness of the methodology is further demonstrated by a sensitivity study that shows how changing the temperature impacts model performance. By combining Hard Negative Mining and temperature parameters, the contrastive learning mechanism in this study optimizes semantic embedding, making the text classification model more precise and efficient in processing complex semantics. Hard Negative Mining (HNM) improves feature representation in contrastive learning by emphasizing the most difficult negative samples, which are similar to positive samples but belong to distinct classes. This method drives the model to improve its feature space and learn smaller distinctions between comparable cases, hence increasing the discriminative strength of the learned representations. In the study, HNM is integrated into the contrastive learning framework to improve semantic embeddings and generalization capacity. HNM accelerates model convergence and reduces overfitting by prioritizing tough negative samples over easy ones, resulting in better accuracy and robustness, particularly for complex tasks such as text classification. These strategies have improved classification and generalization ability, especially on diverse text datasets.

3 Evaluation and experiment of the english text classification model

3.1 Experimental design

This experiment aims to explore the performance of the English text classification model that integrates graph neural networks and contrastive learning. The public "20 Newsgroups" dataset is selected for testing. This dataset contains various news articles and can fully demonstrate the model's performance after preprocessing. To evaluate the model, indicators such as accuracy, recall, and F1 value are used to comprehensively measure the classification effect. At the same time, compared with the CNN-based classification model, the advantages of the new method are highlighted, verifying the effectiveness of the combination of graph neural networks and contrastive learning. Through this comparative experiment, the performance improvement of the proposed model and its potential in practical applications can be demonstrated. The experimental environment of this article is shown in Table 1:

Table 1: Experimental environment

Serial Number	Experimental Environment	Specific Configuration
1	Experimental System	Windows 11
2	Programming Language	Python
3	Central Processing Unit	Intel i7, 8 cores
4	Operating Medium	Pycharm
5	Memory	32GB
6	Video Memory	12GB
7	CUDA (Compute unified device architecture) version	11.4
8	GPU Floating Point Computing Power	Single precision 15.7, TFLOPS
9	GPU (Graphics Processing Unit)	NVIDIA GTX
10	Deep Learning Framework	PyTorch
11	database	MySQL

3.2 Experimental results

3.2.1 Accuracy

Accuracy is the key to evaluating model performance. This article compares the accuracy of 15 model tests using these two methods. Figure 4 shows the findings:

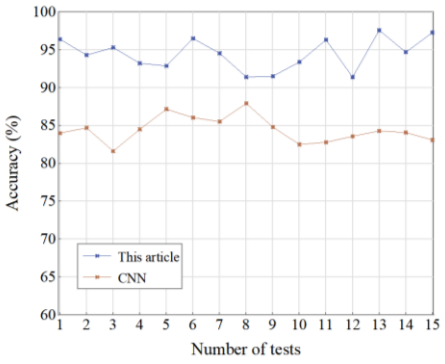

Figure 4: Comparison of model accuracy results under the two methods

According to the 15 test results in Figure 4, the accuracy of this article's method is stable and high, ranging from 91.41% to 97.60%, with an average of 94.46%. The accuracy of the CNN method ranges from 81.63% to 87.94%, with an average of 84.45%. For example, in the first test, this article's method is 12.39% higher than CNN. Even in the eighth test, this method is still ahead. These data prove the advantages of this article's method in dealing with complex semantics and long-distance dependencies. They can distinguish texts more precisely, showing their good generalization ability and robustness. This again proves the effectiveness and superiority of combining graph neural networks with contrastive learning.

3.2.2 Recall rate

The recall rate is the core indicator for evaluating the model's ability to identify positive samples. It reflects the model's ability to find actual positive examples, which is crucial to preventing the omission of key information. A high recall rate means the model can more comprehensively identify relevant text categories, which is particularly important for information retrieval and sentiment analysis tasks because it can reduce underreporting. Based on this, the recall rate of the model is further tested, and the results are shown in Figure 5.

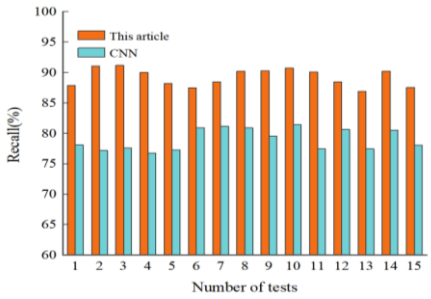

Figure 5: Comparison of recall results under two methods

According to Figure 5, compared with CNN, the recall rate of this article's method is significantly higher, ranging from 86.89% to 91.20%, with an average of 89.27%, while that of CNN is 76.80% to 81.43%, with an average of 79.02%. In the third test, the recall rate of this article's method is 13.61% higher than that of CNN. Even in the 13th test, this method is still ahead. This indicates that the method proposed in this article can more comprehensively recognize text and reduce false negatives. When dealing with imbalanced data, stronger detection of minority categories enhances system reliability. This proves that combining GNNs and contrastive learning can effectively improve recall rates, enhance classification performance, and provide application guarantees.

3.2.3 F1 Value

F1 score is a key indicator for evaluating model performance, which comprehensively reflects the classification performance of the model by combining precision and recall. Optimizing the F1 value can ensure that the model is more accurate and reliable when dealing with imbalanced datasets, reducing misjudgments and omissions. This article calculates the F1 value, as displayed in Table 2.

Table 2: Comparison of F1 value results

Number of tests	This article (%)	CNN (%)
1	91.65	81.00
2	92.69	80.76
3	93.21	79.98
4	91.59	80.53
5	90.44	82.08
6	91.81	83.81
7	91.37	83.34
8	90.82	83.87
9	90.92	81.65
10	92.04	81.97
11	93.08	80.04
12	89.92	82.09
13	92.18	80.71
14	92.42	82.18
15	92.68	80.94

According to Table 2, the F1 value range of this method is 89.92%-93.21%, with an average of 91.79%. The F1 score of CNN ranges from 79.98% to 83.87%, averaging 81.66%. In the third test, the F1 value of this article's method is 13.23% higher than that of CNN. The 12th test also shows that this method is better than CNN. This shows that this method is accurate and reliable, can effectively identify positive examples, and is suitable for information retrieval and sentiment analysis tasks. When dealing with unbalanced data, this article's method reduces misjudgments and positive example omissions greatly improves the robustness and practicality of the system, and once again proves the advantages of combining graph neural networks with contrastive learning.

4 Conclusions

This study combines GNN with contrastive learning to innovate the English text classification model. GNN precisely captures the deep relationship between words in the text, while contrastive learning strengthens semantic embedding and improves the model's ability to identify different texts. The experimental results show that the accuracy, recall rate, and F1 value of the new model on the public dataset are better than the traditional CNN model, showing excellent classification performance. This model is more accurate and stable when dealing with complex semantics and long-distance dependencies, opening up new avenues for English text classification. By displaying text through graph structures, the model reveals the associations between words more deeply, while contrastive learning enhances feature representation, making the model better at identifying text categories. This improves classification accuracy and enhances the model's generalization ability and robustness, making it suitable for various application scenarios. However, there are still limitations to this study. The model needs to adjust parameters for specific text classification and is sensitive to hyperparameters, requiring careful tuning. Meanwhile, the unsupervised learning performance also needs to be improved. The combination of GNN and contrastive learning has brought breakthroughs in natural language processing, with broad application prospects in information retrieval, sentiment analysis, and other areas.

Funding

This research is supported by the China Vocational Education Association of Zhejiang Province (Grant No. ZJCV2024C01).

Data availability

All data generated or analyzed during this study are included in the manuscript.

Author contributions

Chen Sia, Pan Guoqiang is contributed to the design and methodology of this study, the assessment of the outcomes, and the writing of the manuscript.

References

[1] Martinez-Rodriguez, J. L., Hogan, A., & Lopez-Arevalo, I. (2020). Information extraction meets the semantic web: A survey. *Semantic Web, 11*(2), 255–335. https://doi.org/10.3233/SW-180333

[2] Tamine, L., & Goeuriot, L. (2021). Semantic information retrieval on medical texts: Research challenges, survey, and open issues. *ACM Computing Surveys, 54*(7), 1–38. https://doi.org/10.1145/3462476

[3] Martinez-Rodriguez, J. L., Lopez-Arevalo, I., & Rios-Alvarado, A. B. (2022). Mining information from sentences through Semantic Web data and Information Extraction tasks. *Journal of Information Science, 48*(1), 3–20. https://doi.org/10.1177/0165551520934387

[4] Gao, L., Zhang, L., Zhang, L., & Huang, J. (2022). RSVN: A RoBERTa sentence vector normalization scheme for short texts to extract semantic information. *Applied Sciences, 12*(21), 11278. https://doi.org/10.3390/app122111278

[5] Gharagozlou, H., Mohammadzadeh, J., Bastanfard, A., & Ghidary, S. S. (2023). Semantic relation extraction: A review of approaches, datasets, and evaluation methods with looking at the methods and datasets in the Persian language. *ACM Transactions on Asian and Low-Resource Language Information Processing, 22*(7), 1–29. https://doi.org/10.1145/3588940

[6] Yu, S. (2024). Extraction and analysis of semantic features of English texts under intelligent algorithms. *Automatic Control and Computer Sciences, 58*(1), 109–115. https://doi.org/10.3103/S0146411624010123

[7] Wang, K., Ding, Y., & Han, S. C. (2024). Graph neural networks for text classification: A survey. *Artificial Intelligence Review, 57*(8), 190. https://doi.org/10.1007/s10462-023-10290-1

[8] Zong, D., & Sun, S. (2022). Bgnn-xml: Bilateral graph neural networks for extreme multi-label text classification. *IEEE Transactions on Knowledge and Data Engineering, 35*(7), 6698–6709. https://doi.org/10.1109/TKDE.2022.3140011

[9] Vo, T. (2022). An integrated topic modelling and graph neural network for improving cross-lingual text classification. *ACM Transactions on Asian and Low-Resource Language Information Processing, 22*(1), 1–18. https://doi.org/10.1145/3530800

[10] Deng, Z., Sun, C., Zhong, G., & Mao, Y. (2022). Text classification with attention gated graph neural network. *Cognitive Computation, 14*(4), 1464–1473. https://doi.org/10.1007/s12559-021-09960-1

[11] Parthasarathy, K. (2023). ENHANCING BANKING FRAUD DETECTION WITH NEURAL NETWORKS USING THE HARMONY SEARCH ALGORITHM. International Journal of Management Research and Business Strategy, 13(2), 34-47.

[12] Salle, A., & Villavicencio, A. (2023). Understanding the effects of negative (and positive) pointwise mutual information on word vectors. *Journal of*

Experimental & Theoretical Artificial Intelligence, 35(8), 1161–1199. https://doi.org/10.1080/0952813X.2023.2172065

[13] Yao, M., Zhuang, L., Wang, S., & Li, H. (2022). PMIVec: A word embedding model guided by pointwise mutual information criterion. *Multimedia Systems, 28*(6), 2275–2283. https://doi.org/10.1007/s00530-022-00912-3

[14] Hong, D., Gao, L., Yao, J., Zhang, B., Plaza, A., & Chanussot, J. (2020). Graph convolutional networks for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing, 59*(7), 5966–5978. https://doi.org/10.1109/TGRS.2020.3026211

[15] Kazi, A., Cosmo, L., Ahmadi, S. A., Navab, N., & Bronstein, M. M. (2022). Differentiable graph module (DGM) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 45*(2), 1606–1617. https://doi.org/10.1109/TPAMI.2022.3140011