TSS-MLGNN: A Multi-Level Graph Neural Network Approach for Text Semantic Similarity Computation

Juanjuan Li*, Xiaojing Kong
Xuchang Vocational Technical College, Xuchang 461000, China
E-mail: lijuanjuan8211@163.com, xckxjing@163.com
*Corresponding author

Keywords: GNN, text semantic similarity, multi-level graph learning, Feature fusion, attention mechanism

Received: June 10, 2025

In response to the challenges of short text noise and complex semantic associations in online social networks and media, a new method for calculating text semantic similarity is proposed. The purpose of this method is to overcome the limitations of traditional methods in identifying deep semantic relationships, thereby improving accuracy and efficiency. This method proposes a multi-level graph representation learning method based on graph neural network, and establishes a text semantic similarity calculation model based on graph neural network. The graph neural network is used to dynamically learn the complex relationship between nodes, and the multi view semantic features are extracted in parallel with the multi head attention mechanism. Finally, the hierarchical aggregation strategy is used to generate a high-order graph embedding representation that integrates local details and global semantics. Experiments were conducted on two common benchmark datasets, the semantic text similarity benchmark (STS-B, including subsets a-d) and the quora question pairs dataset (QQPD). Performance was evaluated using multiple metrics including accuracy, F1 score, Pearson correlation, calculation time, and memory consumption. Compared to baseline models such as Roberta, Siamese LSTM, and Ernie, the proposed model achieved the highest prediction accuracy. It had F1 scores of 94.21% and 91.38%, respectively, as well as a Pearson correlation coefficient of 0.85. The calculation time and memory consumption were reduced by 26.21% and 13.55% on average. These results demonstrate the effectiveness and robustness of the method in capturing fine-grained and long-distance semantic dependencies, particularly in noisy and informal social media environments. The method is highly applicable to real-time information review and rumor detection. It provides a more accurate computing tool for semantic analysis tasks in online social networks and media scenarios, and has a wide range of practical value in applications such as real-time information review, rumor detection and recommendation systems.

Povzetek: Prispevek se ukvarja z analizo semantične podobnosti kratkih, hrupnih besedil v družbenih omrežjih, kjer klasični modeli ne zaznajo globokih odvisnosti. Predlaga TSS-MLGNN, večnivojski grafni model z GNN, multi-head pozornostjo in hierarhično agregacijo.

1 Introduction

Online social networks and media (OSNEM) have become the primary platform for global information sharing and social interaction. The massive, heterogeneous data generated by OSNEM users creates new demands for understanding text semantics. Calculating text semantic similarity (TSS) is one of the key jobs in natural language processing (NLP). It can be used in a variety of domains, including knowledge management, data mining, recommendation systems, and information retrieval. With the rapid development of the Internet, TSS computation as a core task faces challenges such as complex text data structure and multidimensional semantic relations, which makes efficient and accurate computation of TSS an urgent need [1-2]. The structure of social media text is complex and its characteristics are variable. Traditional text similarity calculation methods find it difficult to accurately portray its semantic relationships. Social media text is highly fragmented and context dependent. The dissemination of information is accompanied by emotional tendencies and implied semantics, which increases the difficulty of calculating semantic similarity. However, semantic similarity-based computation plays an important role in information recommendation systems, public opinion analysis, and the detection of false information. It also involves the problem of cross-modal information fusion. In the face of the characteristics of social media texts, such as fast real-time updating, changing language styles, and huge data size, it is significant to construct an efficient and robust TSS computation model. Traditional statistical or shallow neural network-based methods are difficult to effectively capture the deep lexical, syntactic, semantic, and other multi-level associations in text, resulting in missing information and bias in semantic representation [3]. Bag-of-words models and rule-based, traditional text similarity calculation methods, such as the literal

distance metric, the vector space model, and the latent semantic analysis model, have obvious deficiencies in semantic understanding. These deficiencies lead to inaccurate calculation results when dealing with texts containing polysemous words or contextual associations [4]. Because of its benefits in handling unstructured graph data, graph neural networks (GNNs) have been increasingly popular in text semantic modeling in recent years. However, most of the existing studies are limited to single-view graph structures, such as syntactic dependency or semantic co-occurrence, which makes it difficult to comprehensively integrate the multi-level features of text. In addition, the traditional GNN model has the problem of trade-off between computational efficiency and representation ability when dealing with long-distance dependencies and dynamic semantic associations, and is unable to fully explore the higher-order semantic features of text [5-6]. Based on this, the study proposes TSS computation based on GNN multi-level graph representation learning (TSS-MLGNN) model. The model learns the lexical, syntactic, semantic and other multi-dimensional graph structures of the text in parallel by constructing a multi-head map feature extraction (FE) module and a cross-layer feature fusion mechanism. It also combines graph sampling and attention mechanism to dynamically aggregate neighborhood information, effectively capturing the global semantic dependencies and local fine-grained features of the text. It aims to break through the limitations of semantic modeling from a single perspective and provide a new solution for the accurate measurement of semantic relationships in complex texts. The innovation of the study is that it proposes a multi-level graph representations (MLGR) learning method to comprehensively capture the semantic features of text, which provides a new direction for the computation of TSS.

2 Related works

As a powerful learning method for graph data representation, GNN has been widely used in recent years in the fields of NLP, computer vision, and recommender systems. To improve the out-of-distribution generalization ability of GNNs, Li et al. proposed an out-of-distribution generalized GNN. The network's out-of-distribution generalization ability was improved by utilizing random Fourier features and iteratively optimizing sample graph weights and graph encoders. This provided a robust solution for dynamic data scenarios, such as those involving wearable medical devices [7]. To resolve the problem of high consumption of computational resources and the inability to effectively extract different dimensional correlations in hyperspectral graph classification by traditional GNNs, Li W et al. proposed a dual-stream GNN fusion network. This method realized high-performance and low-computational cost classification through sub-cube input and spatial and spectral branch FE and fusion [8]. Rong C et al. suggested a GNN-based knowledge reasoning technique to overcome the difficulties presented by quickly shifting environments and unanticipated dangers in UAV mission planning systems. By collecting temporal dynamics and structural information, the technique was able to accurately identify missing entities or relationships in the knowledge graph, improving mission planning efficiency [9]. To address the communication overhead bottleneck of GNN training on huge graph datasets, Wang Y et al. proposed a coded neighbor sampling framework. The method reduced the communication overhead by combining the coding technique with the GNN sampling method and utilizing the data excess property. Experimental results demonstrated that the framework significantly reduced the communication overhead, decreased the running time and improved the throughput [10].

The development of deep learning (DL) technology in recent years has given rise to fresh concepts for text similarity calculations. Zhao Y et al. suggested a way to create a fusion matrix utilizing text similarity metrics in order to address the limitations of semantic categorization of medical texts. The efficacy of the fusion matrix in academic text clustering was confirmed by experiments that showed the system performed exceptionally well on text grouping and that the high-frequency words could successfully identify the category meanings [11]. Majumder G et al. developed an interpretable semantic text similarity approach based on block alignment to evaluate the quality of summaries and proposed a deep neural network-based abstract text summarizing method to handle the growing volume of text data. This study achieved efficient and interpretable text summarization generation and evaluation through these two approaches [12]. To overcome the challenges of Chinese semantic complexity and ambiguity in text similarity modeling, Huang J et al. proposed a DL-based computational model for semantic text similarity. The method improved the accuracy of big data analysis by combining character and word granularity embedding, using bidirectional simple recursive units and soft-aligned attention techniques, as well as integrating improved convolutional neural networks [13]. To improve the accuracy of semantic similarity calculations, Li M et al. proposed a DL model based on a Transformer bidirectional encoder. They achieved dual improvements in similarity calculation accuracy and efficiency by using bidirectional encoder embeddings and multi-head attention layers to extract semantic features and by recalling similarity issues through edit distance [14]. To improve the match between job seekers and job descriptions, Chen Y proposed a model based on character embeddings, bidirectional LSTM, and attention mechanisms. The model significantly improved recommendation accuracy by pre-training Chinese text, enhancing contextual understanding, and calculating short text similarity. This provided efficient technical support for human resources recommendations [15]. In recent years, improvements to vector space models for

specific fields (such as news retrieval) have also been proposed. For example, Xiong et al. improved the calibration rate and query speed of news retrieval by dividing news feature words into four semantic categories (time, place, person, and event) and constructing independent vector spaces to calculate similarity [16].

To summarize, most of the existing text similarity studies focus on matching at the lexical level, ignoring the deep semantic information and multi-level connections of text. The performance comparison of existing mainstream TSS calculation methods is shown in Table 1.

In Table 1, while several methods achieve reasonable performance, most are either domain-specific or fail to capture deep semantic associations in short, informal texts. In contrast, the proposed TSS-MLGNN addresses these limitations through multi-level graph learning and multi-head attention, offering improvements in accuracy, robustness, and computational efficiency across diverse datasets. Therefore, the study constructs the TSS-MLGNN model based on the consideration of the multidimensional characteristics of text semantic associations in order to realize the accurate and efficient calculation of TSS.

Table 1: performance comparison of existing mainstream TSS calculation methods

Reference	Method type	Accuracy (%)	F1 score (%)	Pearson corr.	Computation time (s)	Notable limitation
[11] Zhao et al. (2023)	Fusion matrix + similarity metric	85.42	84.17	0.78	0.85	Lacks deep semantic modeling
[12] Majumder et al. (2024)	Interpretable semantic block alignment	87.29	86.15	0.8	0.67	Limited to summary evaluation
[13] Huang et al. (2022)	SRU+CNN hybrid	90.63	90.63	0.82	1.36	Weak for informal text
[19] Viji et al. (2022)	Fine-tuned BERT+BiLSTM	88.15	87.29	0.8	0.46	High memory consumption
Ours (TSS-MLGNN)	GNN+MLGR+multi-head attention	94.21	94	0.85	0.29	/

3 Methods and materials

3.1 Design of GNN-based learning method for MLGR

Although user-generated text data from online social networks, such as tweets and comments, contains rich semantic information and diverse attribute types, it also faces challenges such as noise interference from acronyms and emoticons, as well as structural complexity. Text data often has a complex structure and diverse types of attributes, including lexical, syntactic, semantic, and other levels. To realize the design of TSS computation method, the study firstly designs a GNN-based graph representation learning method by taking advantage of GNN's advantage in processing complex graph structure data. The text's lexical, syntactic, and semantic information is represented by the relationships between nodes and edges. Graph representation learning transforms text data into a low-dimensional, dense vectorized representation. Meanwhile, the semantic information and structural features of the text data are preserved, which in turn are used to calculate the semantic similarity between texts. Figure 1 illustrates the GNN-based graph representation learning technique. As shown in Figure 1, the graph representation learning method based on GNN uses a graph attention network as its core filtering component.

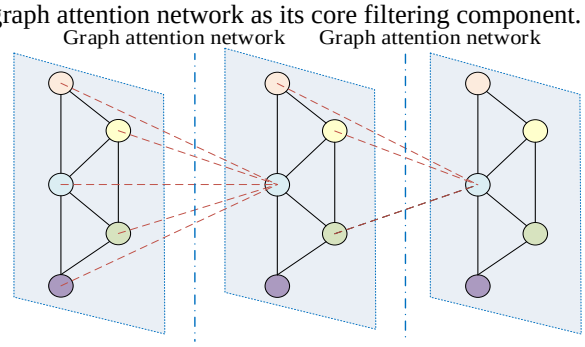


Figure 1: GNN-based approach to graph representation learning

The "graph filtering layer" is a filtering module that uses graph attention mechanisms or graph convolutions. It is used for local smoothing and extracting node feature information from the graph structure. The graph filtering layer consists of a series of graph attention network sublayers. The first layer uses the SpaCy Dependency Syntax Analyzer to transform textual data into graph structures. This process accurately captures the dependencies between individual words in a sentence and provides structured information for subsequent graph construction. Text graphs are constructed through dependent syntactic analysis or semantic association. In these graphs, words or phrases serve as nodes, and syntactic or semantic relations serve as edges. Each node is assigned a feature representation using GloVe word embeddings as an initial representation. The second layer uses a GNN to learn deep features from the graph structure, aggregate information from neighboring nodes via a message passing mechanism, and dynamically update node representations to capture global semantic dependencies in text. Ultimately, through hierarchical graph representation learning, the model can extract higher-order semantic features of the text and provide robust vectorized representations for the similarity computation task. In addition, when constructing the co-occurrence graphs, a sliding window approach is used with a window size of 5 to capture contextual information between words. However, considering that the semantic associations of text have multi-dimensional characteristics such as syntax, co-occurrence, context, etc., relying only on a single perspective is prone to ignore other potential associations, resulting in missing or biased information. Therefore, the study combines multi-level features for optimization on this basis, and learns the node relationships of each perspective separately by processing different graph structures in parallel and using the multi-head attention mechanism of graph attention network. For the different types of syntactic and co-occurrence relations, the study devises a multiplicative strategy for different edge types in the graph construction process. Specifically, syntactic and co-occurrence relations are modeled by independent graph representations, and each relation type corresponds to an independent edge type. Then, the graph neural network processes these edge types separately. It does so by training different graph structures in parallel. Finally, it fuses these different perspectives at the feature fusion stage of each graph layer. This allows it to capture the text's multilevel semantic relations. This approach preserves multiple semantic relations, such as syntax and co-occurrence. It also effectively avoids information loss, improving the accuracy and robustness of TSS computation. Meanwhile, the graph sampling and aggregation algorithm is used to hierarchically sample neighboring nodes and iteratively aggregate local features, so as to more accurately measure the global semantic similarity among texts [17-18]. The optimized GNN-based MLGR learning method is shown in Figure 2.

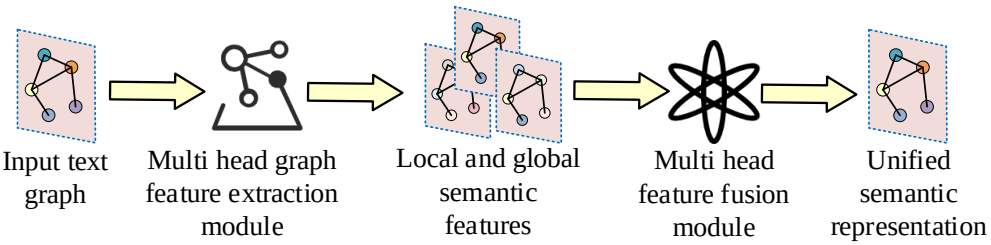


Figure 2: GNN-based approach to learning MLGR

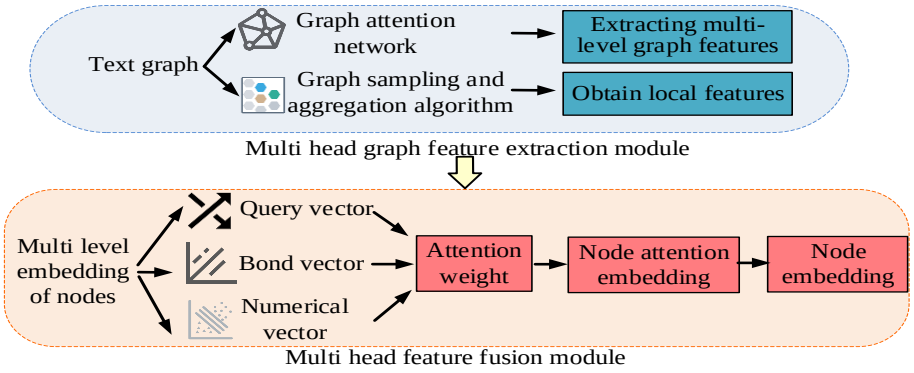


Figure 3: Basic structure of multi-head map FE module and multi-head feature fusion module

In Figure 2, the GNN-based MLGR learning method designed in the study consists of two parts: multi-head map FE module and multi-head feature fusion module. Through a parallel multi-head attention mechanism, the former dynamically learns the association weights between nodes of different levels

(such as lexical, syntactic, and semantic) to capture the local and global differentiated semantic features of the text graph, respectively. The latter performs feature fusion on the outputs of the multi-heads to integrate the scattered multi-view information into a unified semantic representation. Through the combination of hierarchical filtering and cross-head fusion, the fine-grained semantics of the critical paths in the text graph can be preserved while suppressing noise interference. The final generated higher-order graph embeddings can accurately reflect the deep semantic associations among texts and provide robust feature support for similarity computation. Among them, the basic structure of multi-head map FE module and multi-head feature fusion module is shown in Figure 3.

In Figure 3, the multi-head map FE module extracts map features from different viewpoints by receiving input maps and extracting them separately through multiple map attention networks, while sampling and aggregating the map structure using the map sampling and aggregation algorithm to obtain local features. By capturing the complex relationships and patterns in the graph data, it provides rich information for subsequent feature fusion. The multi-head graph feature fusion module, on the other hand, is responsible for receiving node multi-level embeddings from the multi-head FE graph module. Then, the node attention embeddings are generated through the computation of query vectors, key vectors, and value vectors to generate the attention weights for weighting the features in different viewpoints. Finally, the final node embedding representation is obtained by node attention embedding computation. In this instance, Equation (1) displays the calculation of the query, key, and value vectors.

$$\begin{cases} Q = W_Q X \\ K = W_K X \\ V = W_V X \end{cases} \tag{1}$$

In Equation (1), Q represents the query vector, which is used to retrieve relevant information from all keys. K represents the key vector, which is used to calculate the similarity with the query vector and measure the "matching degree" of each position to the current query. V represents the numerical vector, which carries the actual information to be aggregated and is ultimately output as a weighted sum based on the attention weight. W_Q, W_K, and W_V denote trainable projection matrices that map the inputs to query, key, and value spaces, respectively. X denotes the input matrix. The attention scores are computed by scaling dot product and normalized to weight distribution as shown in Equation (2) [19].

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)\cdot V \tag{2}$$

In Equation (2), $\text{Attention}(Q,K,V)$ denotes the attention weight. d_k indicates the dimension of key vector K. $\sqrt{d_k}$ denotes the scaling factor. The semantic associations between nodes or words are captured by Equation (2) to generate higher-order feature representations. In the multi-head mechanism, each head is computed independently, as shown in Equation (3).

$$\text{head}_i = \text{Attention}(X\cdot W_i^Q, X\cdot W_i^K, X\cdot W_i^V) \tag{3}$$

In Equation (3), head_i denotes the attention output of the i th head. W_i^Q, W_i^K, and W_i^V denote the exclusive projection matrices of the i th head, respectively. Equation (3) calculates the multi-head output fusion, which is displayed in Equation (4).

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\ldots,\text{head}_h)\cdot W^O \tag{4}$$

In Equation (4), $\text{MultiHead}(Q,K,V)$ denotes the output of multi-head attention. Concat denotes the connectivity function. head_1 and head_h denote the output of the 1st head and the output of the h th head, respectively. W^O denotes the output projection matrix for compressing the spliced multi-level features back to the original dimension.

3.2 TSS-MLGNN method design

On the basis of GNN MLGR learning method based on GNN, the study proposes the TSS-MLGNN method model. Based on the GNN MLGR learning method can transform text data into graph structures, capture the structural information of text through multi-level learning, and understand the syntactic and semantic structure of text. Based on this, a TSS computation model is constructed to accurately capture text semantic information, effectively process relational structures, and improve the accuracy and robustness of similarity computations. A strategy combining grid search and random search is adopted to explore the impact of different hyperparameter combinations on model performance. During hyperparameter tuning, a validation set is used to evaluate the effectiveness of different combinations of hyperparameters. The validation set is a subset of the training set, which ensures that the model does not overfit during training. Cross-validation is then used to assess the performance of each set of hyperparameters. The basic structure of TSS-MLGNN is shown in Figure 4.

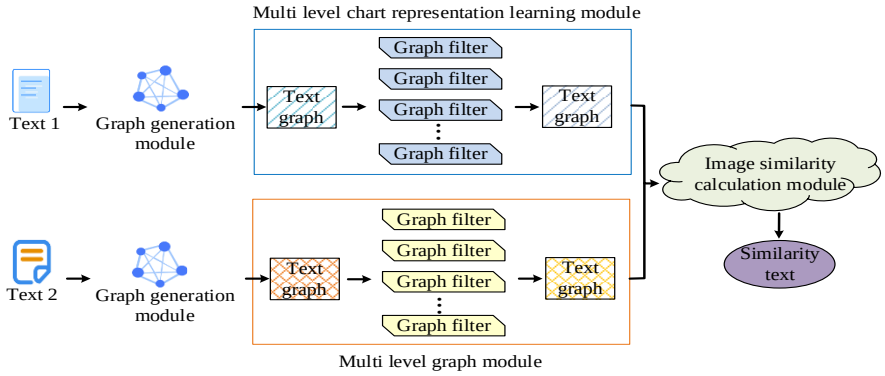


Figure 4: Basic structure of the TSS-MLGNN

In Figure 4, the TSS-MLGNN model proposed in the study contains three main parts: graph generation module, MLGR learning module, and graph similarity calculation module. In this method, short text data is processed by constructing a MLGR graph learning module to capture different aspects of the text. The graph generation module is responsible for converting the text into a text graph. Among them, nodes represent words or phrases in the text and edges represent relationships between them. Each text graph is then fed into multiple graph filters that work in parallel to extract features of the graph from different perspectives. The output of the graph filters is fed to the graph similarity calculation module, which is responsible for calculating the similarity scores between graph representations obtained from different viewpoints. It also assesses the similarity between the texts by comparing the features of the graphs output from different graph filters. The basic framework of the graph generation module is shown in Figure 5.

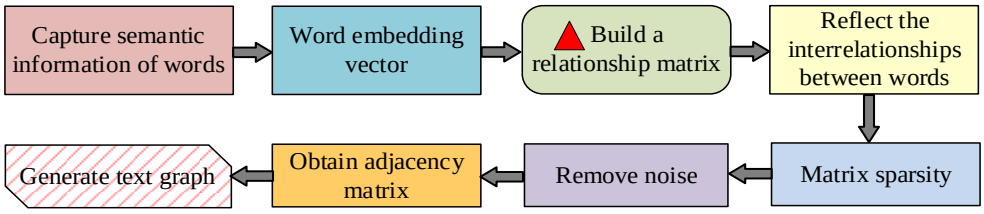


Figure 5: Basic framework of the graph generation module

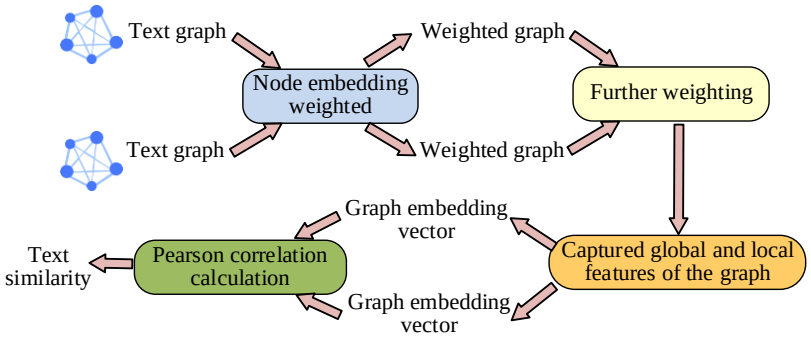


Figure 6: Basic framework of the graph similarity computation module

In Figure 5, the graph generation module designed by the study first converts each word or phrase in the text into a vector representation by means of word embedding vectors, which are used to capture the semantic information of the words. Moreover, these word embedding vectors are utilized to construct a relation matrix to reflect the interrelationships between words in the text. Among them, the construction of the relation matrix is the core of text graph generation, which determines the connection between word nodes. Then, unimportant relations or noise are removed by matrix sparsification to obtain a neighbor matrix. The adjacency matrix (AM) retains only the connection relations between word pairs with strong associations as a way to reduce the computational complexity. Finally, the sparsified matrix is used to generate a text graph. Each word corresponds to a node in the graph. The elements of the AM with non-zero values indicate the existence of edges between the corresponding nodes, i.e., there is an association between words. By generating the graph structure to visually represent the complex relationships of text data, it provides the data basis for the subsequent

task of MLGR learning using GNN. The computation of the relational AM is shown in Equation (5).

$$r_{ab} = (W_h \odot h_a) \times (W_h \odot h_b) \tag{5}$$

In Equation (5), a and b denote the word embeddings of different words, respectively. r_{ab} denotes the inter-word relationship score between word embedding a and word embedding b. W_h represents the strength of the relationship between nodes. It not only projects the original node representation onto a subspace suitable for relationship modeling, but also automatically adjusts the importance of each dimension through the training process to characterize the strength distribution of the relationships between words. In backpropagation, W_h selectively amplifies or suppresses interaction responses based on final classifications or matching losses to achieve more refined relationship modeling. h_a and h_b denote the word embedding vectors of the a th and b th words, respectively. The formula for performing matrix sparsification is shown in Equation (6).

$$r'_{ab} = \begin{cases} r_{ab}, r_{ab} \geq \gamma \\ 0, r_{ab} < \gamma \end{cases} \tag{6}$$

In Equation (6), r'_{ab} denotes the sparsified intermediate matrix. γ represents the sparsification threshold, which is used to regulate the semantic density of the graph structure. Further the AM is generated as shown in Equation (7).

$$\Gamma_{ab} = \begin{cases} 1, r'_{ab} > 0 \\ 0, r'_{ab} \leq 0 \end{cases} \tag{7}$$

In Equation (7), Γ_{ab} denotes the AM, which is used to represent the connection relationship between nodes. The generated text graph structure first enters into the MLGR learning module to complete the FE of the text graph and the mining of the internal semantic structure information of the text. Subsequently, it enters into the graph similarity calculation module to calculate the similarity score between text graphs using the processed text graph features. The details are shown in Figure 6.

In Figure 6, the processed text graph is first generated as a weighted graph through the node embedding weighting process. Each node of the text graph is given a different weight based on its importance or contextual relationship in the text. In this way, the semantic information represented in the graph is enhanced, making the subsequent graph matching process more accurate. Subsequently, the weighted graphs are further processed and each graph is converted into a graph embedding vector. This vector is a compact representation of the graph structure and node features, capturing both global and local features of the graph. Finally, a correlation calculation is performed using Pearson's correlation coefficient to compare the similarity between two graph embedding vectors and obtain the short text similarity score. The formula for similarity is shown in Equation (8).

$$S = \frac{|\delta[(d_1 - \bar{x}_{d_1})(d_2 - \bar{x}_{d_2})]|}{\sqrt{\sum_{k=1}^{n}(d_1^k - \bar{x}_{d_1})^2}\sqrt{\sum_{k=1}^{n}(d_2^k - \bar{x}_{d_2})^2}} \tag{8}$$

Table 2: Pseudocode for the TSS-MLGNN model

Algorithm 1: Training Procedure of TSS-MLGNN
Input: Multi-semantic graphs G = {G_syn, G_coo}, node features X, label set Y, max epochs T, learning rate η
Output: Optimized model parameters θ*
1: Initialize model parameters θ, semantic attention weights α, layer-wise transformation matrix W
2: for t = 1 to T do
3: // Step 1: Graph convolution over each semantic graph
4: for each semantic graph G_l in {G_syn, G_coo} do
5: H_l = GCN(G_l, X; θ_l) // GCN encodes semantic graph G_l
6: end for
7: // Step 2: Semantic-level attention-based fusion
8: for each node i do
9: e_l^i = LeakyReLU(a^T [W·H_l^i || W·H_avg^i]) // Compute attention logits
10: α_l^i = softmax(e_l^i) // Normalize attention scores
11: Z^i = Σ_l α_l^i · H_l^i // Aggregate semantic-level embeddings
12: end for
13: // Step 3: Prediction and backpropagation
14: Y_pred = Softmax(MLP(Z))
15: L = CrossEntropy(Y_pred, Y) + λ ||θ||² // Loss with L2 regularization
16: Update θ using Adam optimizer
17: end for
18: Return optimized parameters θ*

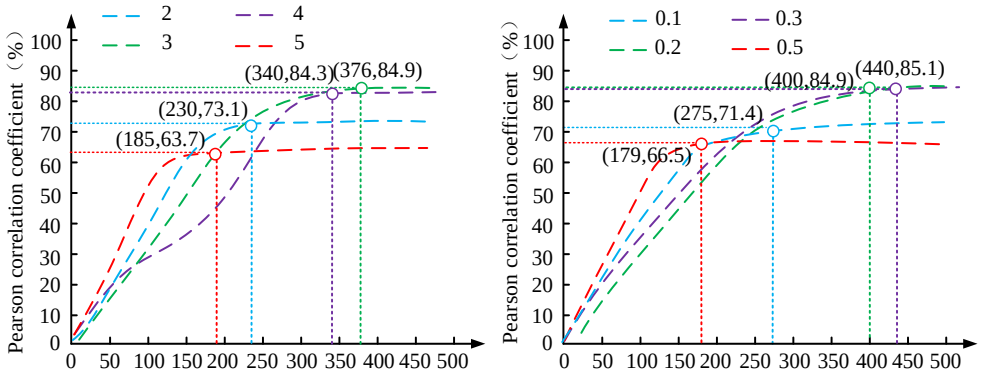
In Equation (8), S denotes the Pearson correlation coefficient between the two variables. δ denotes the expected value. d_1 and d_2 denote map embedding vectors. $\bar{x}_{d_1}$ and $\bar{x}_{d_2}$ denote the mean of d_1 and d_2, respectively. d_1^k and d_2^k denote the values of d_1 and d_2 at the k th position, respectively. n denotes the vector dimension. Through the collaborative work of the graph generation module, the MLGR learning module, and the graph similarity computation module, the TSS-MLGNN model is able to efficiently process textual data and capture the complex relationships and deep semantics in the text. This offers a dependable and efficient way to calculate text similarity in the context of NLP. The pseudocode for the TSS-MLGNN model is shown in Table 2.

4 Results

4.1 Experimental environment and TSS-MLGNN model performance validation

To test the effectiveness of the newly proposed TSS computation method, the study sets up a suitable experimental environment. Windows 11 operating system is used, the algorithm language is Python, and PyTorch 1.12.0 is used as the DL framework. The CPU is Intel® CoreTMi7-9700, the GPU is NVIDIA RTX 3090, and the memory is 64GB. Meanwhile, the learning rate is set to 0.01. The STS Benchmark dataset and Quora Question Pairs dataset (QQPD) are used as the test data sources. ERNIE, Siamese-LSTM, and RoBERTa are retrained and fine-tuned in the same experimental setting using the same data preprocessing procedure, hyperparameter settings, and training cycles. The study first conducts sensitivity selection tests on two types of hyperparameters that have a significant impact on the performance of the final model, namely the number of graph filtering layers l and the sparsity threshold γ. To assess the statistical reliability of the hyperparameter selection, each hyperparameter configuration is run independently five times. The test results are shown in Figure 7.

Figure 7(a) shows the influence curve of different graph filtration layers on the Pearson correlation coefficient. In the preliminary iteration stage, the model with larger graph filtration layers converges significantly faster. As the iteration approaches 376, the model's detection accuracy with graph filtration layers of 3 and 4 increases to 84.3% and 84.9%, respectively. This surpasses the accuracy achieved with layers of 2 and 5. Considering the convergence speed and the balance of correlation, the graph filtration layers of 3 is finally selected for the study. Figure 7(b) shows the influence curve of different sparsification thresholds on the Pearson correlation coefficient. The correlation tends to rise and then fall as the sparsification criteria increase. When the sparsification threshold reaches 0.2, the Pearson correlation coefficient is 84.9%. The correlation coefficient is close to 85.1% when the sparsification threshold is 0.3, and convergence is faster. At the same time, the study calculates the standard deviation of themodel output predictions and the coefficient of variation (CV) of the F1 score under different γ values to assess their stability. The results are shown in Table 3.



(a) Comparison of graph filtration layer test results (b) Comparison of sparsification threshold test results

Figure 7: Sensitivity analysis of the number of graph filtration layers and sparsification thresholds

Table 3: Model stability analysis under different γ values

γ value	AUC mean	Macro-F1	CV (F1)	Predicted standard deviation
0.2	0.78	0.742	0.081	0.026
0.3	0.85	0.774	0.049	0.018
0.4	0.84	0.770	0.056	0.02
0.5	0.81	0.752	0.074	0.025

As shown in Table 3, when γ is set to 0.3, the model achieves optimal performance in terms of AUC, Macro-F1, and other metrics, with low prediction volatility and strong robustness. Therefore, sparsification thresholds of 0.3 are finally selected to ensure efficient and stable model performance. The study continues with ablation tests, comparing the standard GNN text semantic similarity calculation model (TSS-GNN) with a single-head attention mechanism. The ablation experiments are run independently five times on both the STS-B and QQPD datasets, with the results shown in Table 4.

As shown in Table 4, the average Pearson correlation coefficient of the TSS-MLGNN model on the STS-B dataset reaches 0.842, which is significantly higher than the 0.801 of the TSS-GNN model. Additionally, its F1 score and accuracy are 0.928 and

0.918, respectively, both notably superior to the 0.882 and 0.872 of TSS-GNN. This advantage is equally pronounced in the QQPD dataset. There, the TSS-MLGNN model achieves a Pearson correlation coefficient of 0.848, an F1 score of 0.935, and an accuracy of 0.925. Additionally, the TSS-MLGNN model exhibits a smaller standard deviation, indicating lower variability and greater stability in its results. The *p*-values from the paired t-tests are all less than 0.01, further confirming that the performance advantage of TSS-MLGNN is highly statistically significant. This indicates that introducing multi-level graph feature learning and multi-head attention mechanisms significantly improves the model's performance in TSS calculation tasks and enhances its robustness. This enables the model to maintain stable, high performance across different datasets.

4.2 TSS-MLGNN model simulation validation

To verify the practical application of this TSS computational model, the study introduced more advanced algorithms in the field for comparison. They include robustly optimized bidirectional encoder representations form Transformers (RoBERTa), Siamese long short-term memory network (Siamese-LSTM), enhanced representation through knowledge integration (ERNIE). The lossy iteration effect and mean squared error (MSE) of the four models are first tested. Figure 8 displays the outcomes of the experiment.

Table 4: Statistical comparison of the performance of TSS-MLGNN and TSS-GNN ablation models

Dataset	Model	Pearson Corr. (Mean±CI)	F1 Score (Mean±CI)	Accuracy (Mean±CI)	Std. Dev. (Pearson)	*P* value (vs TSS-GNN)
STS Benchmark	TSS-MLGNN	0.842 ± 0.006	0.928 ± 0.005	0.918 ± 0.005	0.005	-
	TSS-GNN	0.801 ± 0.008	0.882 ± 0.007	0.872 ± 0.007	0.008	< 0.01
QQPD	TSS-MLGNN	0.848 ± 0.005	0.935 ± 0.004	0.925 ± 0.004	0.004	-
	TSS-GNN	0.810 ± 0.007	0.895 ± 0.006	0.885 ± 0.006	0.007	< 0.01

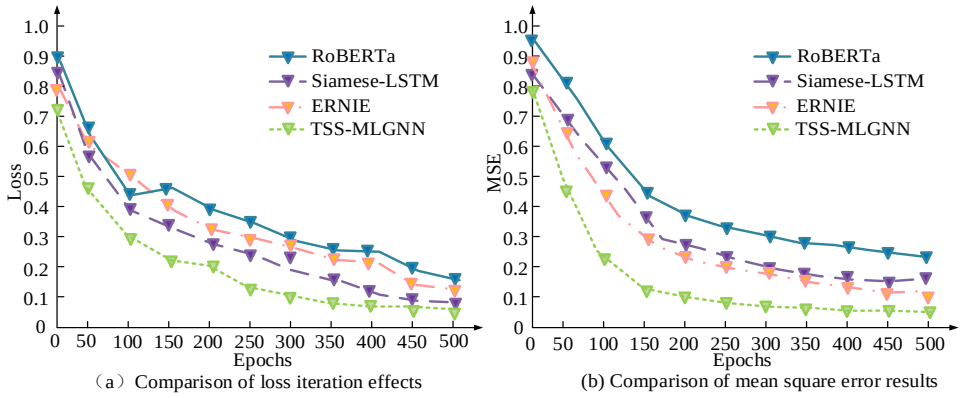


Figure 8: Comparison of loss iteration effect and MSE test results for the four models

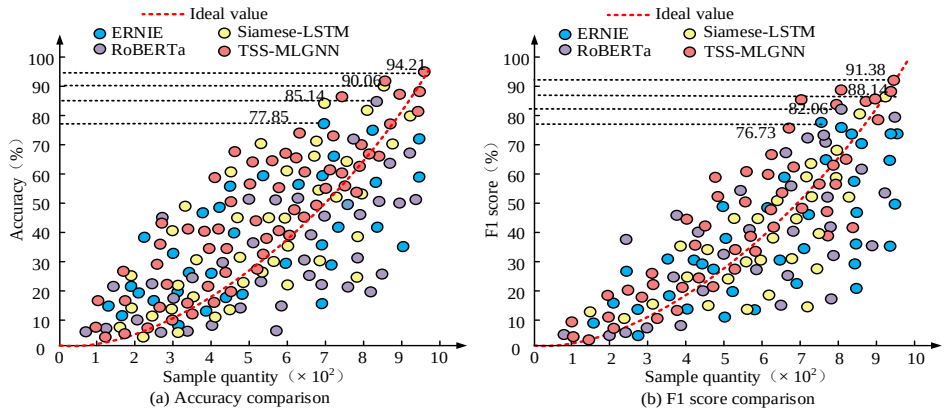
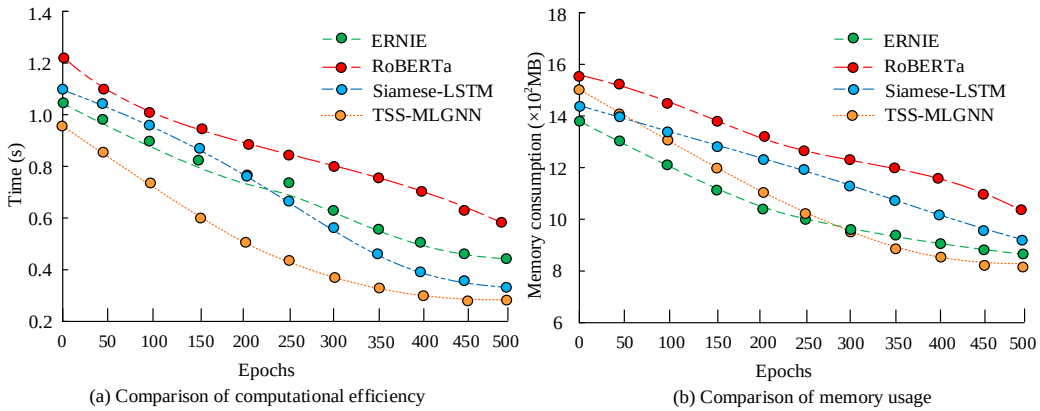


Figure 9: Accuracy and F1 score comparison of the four models

Figure 8(a) shows the curves of the loss values of the four models with the number of training iterations. The TSS-MLGNN model performs the best in terms of loss decrease speed and final results, with significantly lower loss values than the other models. It indicates that GNN-based learning of MLGR can capture the semantic relations among texts faster and has a strong learning ability. In contrast, the Siamese-LSTM model has a relatively slower descent speed although the final loss is also lower, which may be related to its more complex model structure and training process. Meanwhile, in conjunction with Figure 8(b), the MSE value of the TSS-MLGNN model continuously decreases and always stays at the lowest level. When the number of loss iterations reaches 250, its MSE value stabilizes below 0.1. The MSE values of the ERNIE model and the Siamese-LSTM model also perform better, but there is still a gap compared with TSS-MLGNN. Especially in the late stage of training, the MSE value decreases significantly less quickly than the TSS-MLGNN model. Taken together, the TSS-MLGNN model proposed in the study has stronger learning ability and generalization ability in the TSS computation task. Meanwhile, the study tests the accuracy and F1 score of the four models. Figure 9 displays the outcomes of the experiment.

As shown in Figure 9(a), the accuracy changes of the four models under different sample sizes show a curvilinear upward trend. Specifically, the ERNIE model performs the worst, which may be related to the limited adaptation of its knowledge-enhanced pre-training strategy to specific TSS computation. The TSS-MLGNN model performs best at all sample sizes and is closest to the ideal value. With the highest accuracy of 94.21%, it outperforms the other three models by an average of 11.69%. In Figure 9(b), the overall trend of change is similar to the accuracy rate. The TSS-MLGNN model has the highest F1 score of 91.38%, which is an average improvement of 11.02% over the other three models. It shows that GNN-based MLGR learning has a strong advantage for the TSS computation task. On this basis, the study compares the differences in computational efficiency and memory consumption among the four models. Figure 10 displays the outcomes of the experiment.

A comparison of the computation time of text similarity for different models is shown in Figure 10(a). With the increase of the number of iterations, the computation time of various models shows a decreasing trend. The computational time of the TSS-MLGNN model is significantly lower than that of other models. When the number of iterations reaches 500, the computational time of the TSS-MLGNN model is approximately 0.17 seconds. Whereas, the ERNIE model is 0.46 seconds, the Siamese-LSTM model is 0.35 seconds, and the RoBERTa model takes the longest at 0.58 seconds. Compared to other models, the TSS-MLGNN model achieves an average reduction of 26.21%. It shows that the TSS-MLGNN model can effectively reduce the computational overhead when dealing with TSS computation, which is especially suitable for practical applications that need to deal with large-scale data. Combined with Figure 10(b), the memory consumption of the TSS-MLGNN model is large at the beginning of the iteration. However, as the iterations increases, its memory consumption gradually stabilizes and is the lowest. When the iteration reaches 500, the memory consumption of the TSS-MLGNN model is about 824 MB, which is reduced by 13.55% on average compared with other models. It displays that the TSS-MLGNN model has better training optimization and is efficient and practical in TSS computation tasks. Lastly, the study presents the TSS computing techniques suggested by other researchers for comparison in an attempt to further confirm the efficacy of the suggested model. Table 5 displays the comparing results.



(a) Comparison of computational efficiency (b) Comparison of memory usage

Figure 10: Comparison of four models in computational efficiency and memory consumption

Table 5: Performance comparison of different models

Index	Pearson correlation	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)	Computing time (s)	Memory consumption ($\times 10^2$MB)
TSS-MLGNN	0.85	94.21	95.36	94.02	94.00	0.29	8.24
Reference [11]	0.78	85.42	85.78	84.17	84.17	0.85	10.05
Reference [13]	0.82	90.63	91.22	89.94	90.63	1.36	9.84
Reference [19]	0.80	88.15	88.79	87.29	87.29	0.46	11.14

In Table 5, the TSS-MLGNN model performs optimally on all metrics. It is more accurate and successful at capturing TSS, as evidenced by its Pearson correlation coefficient of 0.85, which is an average increase of 6.25% when compared to the other models. However, the TSS-MLGNN model's accuracy, precision, recall, and F1 score are all much higher than those of the other models, at 94.21%, 95.36%, 94.02%, and 94.00%, respectively. In contrast, the Pearson correlation coefficient of reference [11] is only 0.78 and the accuracy is 85.42%, which is a weaker performance. This may be due to the limitations of its model structure or data processing, resulting in its inability to adequately capture the semantic relationships between texts. In addition, the TSS-MLGNN model has obvious advantages in computation time and memory consumption.

4.3 Applicability assessment of TSS-MLGNN model in the field of online social networks and media

As the core carrier of information dissemination and user interaction, OSNEM presents a high degree of heterogeneity, dynamism, and semantic complexity in its user-generated content. To verify the applicability of the TSS-MLGNN model in the OSNEM domain, the study designs systematic experiments to evaluate its performance on typical social media datasets. Two types of typical OSN datasets are selected for the experiments: the Twitter Semantic Similarity dataset (https://developer.twitter.com/en/docs/twitter-api) and the Reddit conversation matching dataset (https://www.reddit.com/dev/api/). Among them, the Twitter Semantic Similarity dataset contains 100,000 pairs of tweets. It is constructed through manual annotation and semi-automatic methods to cover semantic associations between news events and user comments. Three annotators with linguistic backgrounds independently scores the semantic relevance of tweet pairs on a scale of 1 to 5. After removing intermediate-value samples, labels are determined based on the average score ($\geq$3.5 for similar, $\leq$2.5 for dissimilar). A random sample of 10% is used to calculate Krippendorff's alpha coefficient, which is 0.82, ensuring annotation consistency. The Reddit dataset is extracted from the forum, comprising 50,000 comment-reply pairs. Positive samples are high-rated replies reflecting semantic matching, while negative samples are randomly selected unrelated comments. Standardized approval counts are used as auxiliary indicators but are not directly employed for training labels. The test results of the TSS-MLGNN model on the two OSN datasets are shown in Table 6.

Table 6: Test results of the TSS-MLGNN model on the OSN dataset

Dataset	Index	Pearson correlation	Accuracy (%)	F1 score (%)	Computing time (s)
Twitter dataset	TSS-MLGNN	0.82	90.15	94.00	0.21
	RoBERTa	0.75	85.32	84.17	0.38
	Siamese-LSTM	0.68	82.17	90.63	0.45
	ERNIE	0.72	84.56	87.29	0.34
Reddit dataset	TSS-MLGNN	0.79	88.42	86.15	0.24
	RoBERTa	0.73	83.91	81.27	0.41
	Siamese-LSTM	0.65	80.05	78.33	0.49
	ERNIE	0.70	82.64	80.58	0.36

Table 7: Comparison of rumor detection classification effectiveness

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)	AUC-ROC
Siamese-LSTM	83.52	84.17	82.91	82.13	0.879
ERNIE	86.24	87.05	85.62	85.03	0.901
RoBERTa	87.81	88.73	87.35	86.74	0.918
TSS-MLGNN	89.32	90.15	88.93	88.24	0.941

Example 1: Noisy Short Text	Example 2: Semantically Complex Text
Token-level embedding of slang/emojis (e.g., "omg", "😂") → Co-occurrence & dependency parsing → Sparse noise filtering	Word/subword segmentation (EN/CH) → Dependency parsing (e.g., "Despite→glitch") → Cross-lingual edge alignment
Multi-head GAT: Head 1 captures "awesome–cool" similarity; Head 2 models syntax like "so→cool"; Head 3 handles emoji-emotion links	Head 1 captures "surprisingly engaging–attractive" semantics; Head 2 aligns translation pairs; Head 3 models sentiment reversal
Weighted fusion of heads → high-level emotional embedding capturing slang, emoji, sentiment	Attention-based fusion across semantic/syntactic/cross-lingual features → nuanced embedding reflecting turn of sentiment
Pearson correlation ≈ 0.78 → High similarity between joyful reactions	Pearson correlation ≈ 0.90 → Strong semantic match in post-glitch positive sentiment

Figure 11: Step-by-step processing of TSS-MLGNN on noisy and semantically complex texts

According to the experimental results in Table 6, the TSS-MLGNN model shows obvious comprehensive advantages on both Twitter and Reddit, two typical datasets in the field of OSNEM. In the Twitter dataset, the Pearson correlation coefficient, accuracy, and F1 score of TSS-MLGNN are 0.82, 90.15%, and 94.00%, respectively. These scores are, on average, 14.42%, 7.30%, and 7.60% higher than those of the other three models. This indicates that the unique multi-head graph FE module of the TSS-MLGNN model is able to dynamically learn the correlation weights between nodes from different levels, such as lexical, syntactic, and semantic, through the mechanism of multi-head attention. It can comprehensively capture the local and global semantic features of text. Meanwhile, the computation time of TSS-MLGNN model is only 0.21s, which is 46.15% lower than other models on average, indicating that TSS-MLGNN has high efficiency in processing large-scale data. This is attributed to the model's graph sampling and aggregation algorithm. This algorithm reduces unnecessary computational overhead and improves operational efficiency by hierarchically sampling neighboring nodes and iteratively aggregating local features. In the Reddit dataset, the Pearson correlation coefficient, accuracy, and F1 score of TSS-MLGNN are 0.79, 88.42%, and 86.15%, respectively. These scores are, on average, 13.94%, 7.57%, and 7.61% higher than those of the other three models. Although the higher text complexity of Reddit results in a slight increase in model computation time to 0.24s, it is still 42.86% lower than the other models. This indicates that the model is well adapted and efficient in processing on different types of social network text data. Finally, the study evaluates the actual effectiveness of the model in the rumor detection classification task using the PHEME rumor dataset. The results are shown in Table 7.

As shown in Table 7, TSS-MLGNN achieves comprehensive performance leadership in the rumor detection task. Its accuracy rate reaches 89.32%, an improvement of 1.51% over the best baseline RoBERTa. With an F1 score of 88.24%, the model demonstrates an improved balance between precision and recall. Additionally, the AUC-ROC value of 0.941 verifies its ability to accurately distinguish semantic boundaries. This suggests that the Pearson correlation coefficient effectively aids in making binary classification decisions and that the classification strategy based on similarity scores balances semantic sensitivity with engineering practicality.

4.4 Model example analysis

To more intuitively demonstrate how the TSS-MLGNN model processes different types of short texts, the study selects two examples to analyze step by step. One example contains noisy short texts with internet slang and emoticons. The other example contains semantically complex, structurally long mixed Chinese and English texts, as shown in Figure 11.

Figure 11 illustrates the four key steps of the TSS-MLGNN model in processing typical short texts, fully demonstrating its ability to handle semantic noise and complex structures. The model effectively captures deep semantic relationships in texts containing internet slang and mixed languages through FE, graph construction, graph fusion, and semantic enhancement processes. This improves the quality of text representation and the accuracy of semantic matching. This demonstrates its robustness and generalization capabilities in complex contexts.

5 Discussion

The TSS-MLGNN model proposed in the study demonstrated significant performance advantages in several aspects. A comparison with related work in the references indicated that TSS-MLGNN had advantages in terms of task completion, accuracy, and computational efficiency. This suggested that it could handle complex data structures and capture nonlinear relationships more effectively. Specifically, the model proposed in literature [11] demonstrated a more stable accuracy in multiple experiments, but its accuracy improvement was smaller when the data volume increased. Moreover, although the method in literature [13] performed well on small sample

data, its generalization ability on large-scale datasets decreases, and it failed to perform stably in multiple test scenarios. In contrast, TSS-MLGNN maintained high accuracy on large-scale datasets, particularly in capturing complex patterns. This was thanks to its graph neural network architecture, which effectively fused multi-level information and enhanced the model's expressive ability. In terms of model complexity and training time, TSS-MLGNN showed a good balance. Although it had slightly more parameters than the model in the literature [19], its optimized computational structure and efficient graph convolution operation effectively controlled the training time and inference speed. When conducting large-scale experiments, the training time of the TSS-MLGNN model did not increase significantly compared to other models in the literature, but it did demonstrate faster convergence speeds in some experiments. This indicated that TSS-MLGNN was more efficient in the utilization of computational resources and was able to reduce the computational cost while ensuring performance.

The reason for the performance improvement can be attributed to two main aspects. First, the TSS-MLGNN model uses an innovative multilevel graph neural network structure. This structure is more flexible for FE and information transfer. It is also better able to capture local and global dependencies in the data. Second, the TSS-MLGNN model uses an adaptive training strategy during the optimization process. This strategy improves model training efficiency by dynamically adjusting the learning rate and accelerating convergence. These innovative designs enable TSS-MLGNN to achieve better results in several benchmark tasks.

6 Conclusion

With the increasing complexity and diversity of text data, traditional text similarity calculation methods are difficult to adequately capture the deep semantic relationships in text. To overcome this challenge, the study proposed a MLGR learning method based on GNN and constructed the TSS-MLGNN model on this basis. By constructing a multi-level graph structure, the powerful graph representation capability of GNN was utilized to comprehensively capture the syntactic and semantic information in text from multiple dimensions. In the ablation test, the experimental findings demonstrated that the TSS-MLGNN model's Pearson correlation coefficient varied between 83% and 85%, and that its performance was noticeably better than that of the single-view TSS-MLGNN model. Compared with the same type of text similarity calculation models, the TSS-MLGNN model had faster loss iterations and the lowest MSE, which was stable below 0.1. Meanwhile, the accuracy, precision, recall, and F1 score of the TSS-MLGNN model were 94.21%, 95.36%, 94.02%, and 94.00%, respectively. In addition, the computation time and memory consumption of the TSS-MLGNN model decreased by 26.21% and 13.55%, respectively. Furthermore, the Pearson correlation coefficient of the TSS-MLGNN model on the Twitter and Reddit datasets was 0.82 and 0.79, respectively, verifying its efficiency and stability in OSNEM tasks. The results showed that introducing MLGR learning and the multi-head attention mechanism effectively improved the model's stability and generalization ability. Therefore, the TSS-MLGNN model significantly outperformed others in computational accuracy and efficiency and offered a new approach to text semantic analysis. Meanwhile, TSS-MLGNN was an efficient semantic analysis tool for the OSN platform. It could be used in scenarios such as containing rumors and delineating interest communities. The tool had important theoretical significance and practical value. However, the research still has some shortcomings, such as the model's ability to process ultra-long text has not been verified, and the generalizability in multilingual scenarios needs to be further explored. In the future, the study will focus on two things: introducing dynamic graph pruning to optimize processing of long texts and expanding the application scope by combining it with cross-language pre-training. These changes will further enhance the model's practical application value.

Acknowledgement

This study is supported by Exploration and Research on the New "AI+X" Teaching Model for New-quality Skilled Talents Majoring in Data Intelligence (NO: 2024SJGLX0747).

References

[1] Venkanna G, Bharati K F. Improved meta-heuristic model for text document clustering by adaptive weighted similarity. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2023, 31(5): 749-771. https://doi.org/10.1142/S0218488523500451

[2] Li H, Wang W, Liu Z, Niu Y, Wang H, Zhao S, Liu X. A novel locality-sensitive hashing relational graph matching network for semantic textual similarity measurement. Expert Systems with Applications, 2022, 207(1): 1-11. https://doi.org/10.1016/j.eswa.2022.117921

[3] Shala D R, Sheppard-Law S. Measuring text similarity and its associated factors in electronic nursing progress notes: A retrospective review. Journal of Clinical Nursing, 2023, 32(11-12): 2733-2741. https://doi.org/10.1111/jocn.16667

[4] Chen Q, Zhao G, Wu Y, Qian X. Fine-grained semantic textual similarity measurement via a feature separation network. Applied Intelligence, 2023, 53(15): 18205-18218. https://doi.org/10.1007/s10489-023-04277-8

[5] Li R, Cheng L, Wang D, Tan J. Siamese bert architecture model with attention mechanism for textual semantic similarity. Multimedia Tools and

Applications, 2023, 82(30): 46673-46694. https://doi.org/10.1007/s11042-023-14362-5

[6] Pham P, Nguyen L T T, Pedrycz W, Vo B. Deep learning, graph-based text representation and classification: a survey, perspectives and challenges. Artificial Intelligence Review, 2023, 56(6): 4893-4927. https://doi.org/10.1007/s10462-022-10296-5

[7] Li H, Wang X, Zhang Z, Zhu W. Ood-gnn: Out-of-distribution generalized graph neural network. IEEE Transactions on Knowledge and Data Engineering, 2022, 35(7): 7328-7340. https://doi.org/10.1109/TKDE.2022.3183176

[8] Li W, Liu Q, Fan S, Xu C A, Bai H. Dual-stream GNN fusion network for hyperspectral classification. Applied Intelligence, 2023, 53(22): 26542-26567. https://doi.org/10.1007/s10489-023-04456-9

[9] Rong C, Xiaofang D, Lixuan W. GNN-based temporal knowledge reasoning for UAV mission planning systems. Journal of China Universities of Posts and Telecommunications, 2024, 31(1): 12-25. https://doi.org/10.19153/jcupts.2024.31.1.02

[10] Wang Y, Guan T, Niu D, Zou Q, Zheng H, Shi C J R, Xie Y. Accelerating distributed GNN training by codes. IEEE Transactions on Parallel and Distributed Systems, 2023, 34(9): 2598-2614. https://doi.org/10.1109/TPDS.2023.3289137

[11] Zhao Y, Cui L. Fusion matrix–based text similarity measures for clustering of retrieval results. Scientometrics, 2023, 128(2): 1163-1186. https://doi.org/10.1007/s11192-023-04676-8

[12] Majumder G, Rajput V, Pakray P, Bandyopadhyay S, Favre B. Text summary evaluation based on interpretable semantic textual similarity. Multimedia Tools and Applications, 2024, 83(1): 3233-3258. https://doi.org/10.1007/s11042-023-15593-4

[13] Huang J, Ma K. SRU-based multi-angle enhanced network for semantic text similarity calculation of big data language model. International Journal of Information Technologies and Systems Approach (IJITSA), 2022, 16(2): 1-20. https://doi.org/10.4018/IJITSA.310723

[14] Li M, Shen X, Sun Y, Zhang W, Nan J, Zhu J A, Gao D. Using semantic text similarity calculation for question matching in a rheumatoid arthritis question-answering system. Quantitative Imaging in Medicine and Surgery, 2023, 13(4): 2183-2196. https://doi.org/10.21037/qims-22-1065

[15] Chen Y. Human resource recommendation using Chinese BERT, BiLSTM, and short text similarity algorithm. Informatica, 2024, 48(22): 99-111. https://doi.org/10.31449/inf.v48i22.4525

[16] Xiong W. Web news media retrieval analysis integrating with knowledge recognition of semantic grouping vector space model. Informatica, 2024, 48(5): 41-54. https://doi.org/10.31449/inf.v48i5.4313

[17] Sereshki M T, Zanjireh M M, Bahaghighat M. Textual outlier detection with an unsupervised method using text similarity and density peak. Acta Universitatis Sapientiae, Informatica, 2023, 15(1): 91-110. https://doi.org/10.2478/ausi-2023-0006

[18] Yang S, Huang G, Ofoghi B, Yearwood J. Short text similarity measurement using context-aware weighted biterms. Concurrency and Computation: Practice and Experience, 2022, 34(8): 1-11. https://doi.org/10.1002/cpe.6842

[19] Viji D, Revathy S. A hybrid approach of Weighted Fine-Tuned BERT extraction with deep Siamese Bi–LSTM model for semantic text similarity identification. Multimedia tools and applications, 2022, 81(5): 6131-6157. https://doi.org/10.1007/s11042-021-11596-9