Multi-Task English Grammatical Error Detection via BERT-Gram and Syntactic Dependency GNN Fusion

Huihui Lei
Corresponding email: HuihuiLeii@outlook.com
International Education College, Wuchang Institute of Technology, Wuhan, 430065 China

Keywords: BERT-gram, graph neural network, grammatical error detection, multi-task learning, natural language processing

Received: July 18, 2025

With the continuous development of natural language processing technology, grammatical error detection has become a research hotspot. Traditional methods have limitations in complex sentence structure and generalization ability. Therefore, this study proposes a multi-task learning framework for grammatical error detection that integrates BERT-Gram and graph neural networks to improve detection accuracy and efficiency. BERT-Gram is good at semantic understanding, and graph neural networks are good at processing structured data. The combination of the two can give full play to their respective advantages. The proposed multi-task learning framework integrates BERT-Gram with syntactic dependency graph neural networks, leveraging joint optimization of error detection and correction tasks to enhance semantic and syntactic modeling, and validates its effectiveness through F0.5, precision, and recall metrics on benchmark datasets. In this study, we propose a multi-task learning framework for grammatical error detection fusing BERT-Gram and graph neural network, and verify the performance with F0.5 score, precision/recall and other indicators on NUCLE and Lang-8 datasets through joint optimization error detection and correction tasks, so as to realize the synergy between semantic representation of BERT-Gram and GNN syntactic modeling.

Povzetek: Predlagan je večopravilni model za zaznavanje slovničnih napak, ki združuje BERT-Gram in grafne nevronske mreže ter izboljša natančnost zaznavanja.

1 Introduction

As a key topic in natural language processing, grammatical error recognition is closely related to the development of deep learning. In recent years, breakthroughs in pre-trained language models (such as BERT) have injected new vitality into this field [1, 2]. The current mainstream methods mostly adopt the single-task optimization paradigm, which performs well on specific data sets but makes it difficult to capture the deep correlation between language tasks, resulting in the limited generalization ability of the model in complex contexts [3]. In this study, a multi-task collaborative learning system based on the fusion of BERT-Gram architecture and graph neural network is innovatively constructed to enhance the depth of grammar understanding through knowledge transfer between tasks.

Regarding basic model design, Bert-Gram elevates the word-level representation of traditional BERT to the level of the syntactic structure by injecting a syntactic constraint matrix and dependency encoding [4, 5]. At the same time, a dynamic graph attention mechanism is introduced to construct a syntactic dependency graph so the model can explicitly model long-term syntactic relations [6]. This two-way architecture retains the semantic capture advantage of the pre-trained model and enhances the syntactic sensitivity through the structured reasoning ability of the graph network [7]. The framework adopts the hierarchical parameter-sharing strategy to realize the joint learning of syntactic features and semantic representations in the coding layer and differentiated processing through task-specific adapters in the decoding layer. This design synergizes related tasks such as syntax error correction and semantic role labelling [8, 9]. Specifically, we devised a dynamic weighting algorithm grounded on gradient similarity. This algorithm is capable of automatically tuning the loss weight in accordance with each task's learning status, efficiently addressing the prevalent issue of negative transfer in conventional multi-task learning [10].

Under the multi-task learning framework, grammatical error detection is closely linked to other language tasks, such as sentence classification and semantic role labelling. By sharing the parameters of the presentation and task-specific layers, the model can transfer useful information between different tasks and realize knowledge sharing and migration.

A dynamic task weighting mechanism adaptively modulates the contribution of error detection and correction based on training dynamics and task complexity, facilitating balanced optimization. The algorithm employs gradient normalization and uncertainty-based weighting to align multi-task objectives, enhancing generalization across diverse grammatical error types through synergistic interaction

between BERT-Gram semantics and syntactic dependency graph structures.

2 Basic theory and technical depth

2.1 Principles and characteristics of BERT model

The BERT (Bidirectional Encoder Representations from Transformers) model is a pre-trained bidirectional encoder representation based on the Transformer architecture [11, 12]. It is trained on many corpora, forming a bidirectional Transformer model with excellent generalization performance, suitable for various natural language processing tasks [13]. The detailed structure of the BERT model is shown in Figure 1. The multi-head cross-note block is where GNN integration takes place.

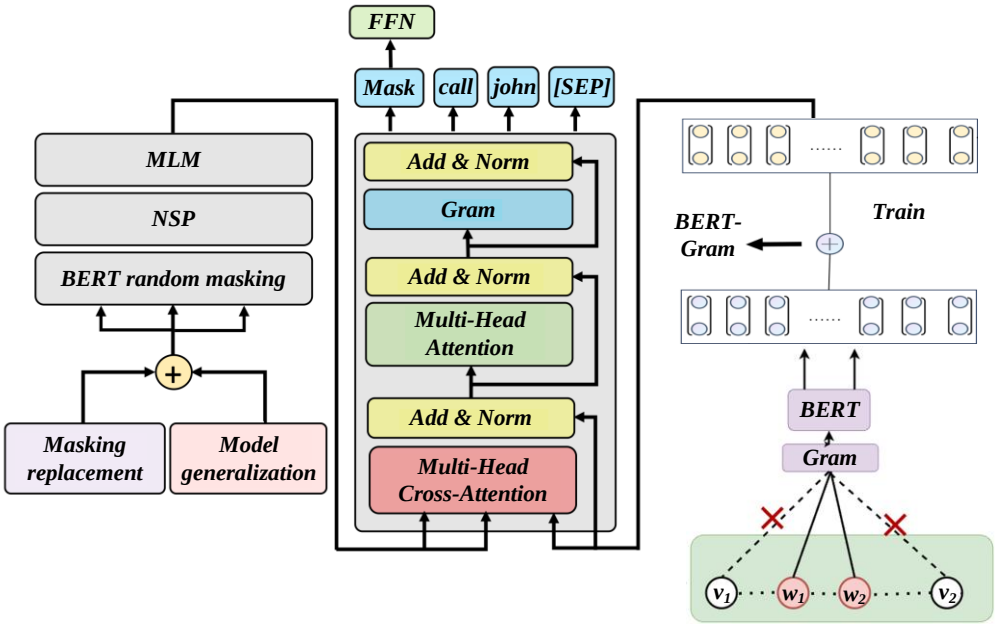

Figure 1: BERT-Gram model architecture

The innovation of the BERT model is mainly reflected in its pre-training process, which includes two core tasks: mask language model (MLM) and next sentence prediction (NSP) [14]. In the MLM task, BERT randomly masks 15% of the words in each sentence and trains the model to predict the masked words according to the context [15, 16]. In order to prevent the model from over-relying on the masking identifier [Mask], 80% probability of masking is replaced with [Mask], 10% probability is replaced with random words, and the remaining 10% is kept unchanged. This design enhances the generalization ability of the model. The NSP task requires the model to determine whether two sentences have a contextual relationship. In the training data, 50% are adjacent sentences, and 50% are randomly combined non-adjacent sentences, aiming to help the model understand the correlation between sentences.

Because BERT uses a large-scale corpus in the pre-training stage, has strong generalization ability, and the output is in vector form, when dealing with specific tasks, it is only necessary to add an output layer based on the pre-training model and make fine-tuning to achieve excellent results without greatly modifying the model structure [17].

2.2 Graph neural network (GNN) basics

Graph Neural Network (GNN) is a neural network that specializes in processing graph structure data [18, 19]. Graphs consist of nodes (vertices) and edges (links) that can represent complex relationships and dependencies [20]. GNN extends traditional neural networks and operates directly on graphs. It can capture graph structural information and learn the representation of nodes or edges, thus performing tasks such as node classification, link prediction, and graph classification [21, 22].

With the development of network technology, the amount of data on the Internet has increased dramatically, and the data forms are diverse [23]. Most data is Euclidean structure data (such as image pixels and sentence words), arranged neatly [24]. However, data such as grammatical structure do not have translation invariance. They are not arranged neatly, which makes it difficult for traditional neural networks to process them effectively, which promotes the development of GNN. GNN learns node-level and graph-level representations by recursively aggregating node neighborhood information, suitable for complex data such as social networks, molecular structures, and knowledge graphs. The current mainstream GNN types include graph convolutional networks (GCN), graph attention networks (GAT), and graph inductive representation learning (GraphSage) [25, 26]. However, GNN still faces challenges in scalability and generalization capabilities, especially when dealing with large-scale graphs and new graph structures.

This study transforms the syntactic dependency tree into a syntactic graph, with each word as a node and the

syntactic arc as an edge. By adding reverse edges and self-ring edges, bidirectional transmission of syntactic information and enhancement of node information are realized [27]. The adjacency matrix A of the graph contains three sub-matrices: Aalong, Arev, and Aloop, which represent the forward edge, the reverse edge, and the self-ring edge, respectively. The syntax graph and word vector are input into the GNN for multi-layer convolution, and the representation of nodes and their adjacent points is obtained, which is used to trigger the word classification task [28].

Table 1 compares the performance of the BERT-Gram and graph neural network-based grammatical error detection multi-task learning framework. Prior approaches to English grammatical error detection, such as BERT-GEC and GECToR, primarily rely on sequence-to-sequence or token-level classification paradigms, often neglecting explicit syntactic modeling. Syntax-aware methods incorporate dependency trees but typically treat syntactic features as static auxiliary inputs rather than dynamic, learnable representations. In contrast, recent advances in graph neural networks (GNNs) enable more flexible syntactic interaction, yet their integration with pretrained language models remains underexplored for error detection. The proposed framework addresses these limitations by unifying BERT-Gram's contextual embeddings with syntactic dependency graphs, fostering deeper interactions between lexical semantics and grammatical structures.

Table 1: GEC methods comparison

Method	Dataset(s)	Key Methodology
BERT-GEC	N/A	Sequence-to-sequence with BERT backbone
GECToR	BEA-2019, CoNLL-2014	Token-level seq2edit with transformers
Syntax-aware CNN/RNN	Custom corpora	Static dependency features + classifiers

3 Design of multi-task learning framework for grammatical error detection based on BERT-Gram and GNN

3.1 Frame overall architecture design

In BERT's initial Mask language model task, by randomly using Mask flags to replace words, the model can learn the meaning of masked words in combination with pre-training context [29]. In this study, homophonic and visually similar characters are innovatively used for masking in MLM tasks to fit the typical characteristics of English grammatical errors. In the task of Chinese grammar error detection, the multi-task learning framework effectively captures the potential associations between different error types by sharing underlying representations, while retaining task-specific layers to achieve fine-grained differentiation. The pre-trained model based on BERT-Gram enhances contextual awareness through sub-word-level semantic encoding, and its output context-related representations form multimodal fusion with the syntactic dependency graph structure based on graph neural networks: the former provides global semantic associations, while the latter explicitly models the grammatical dependencies between words through directed edges. The cross-head attention mechanism plays a key role in this process, with its multi-head parallel computation adaptively focusing on grammatical features in different subspaces and dynamically weighting the importance of dependency paths through query-key value matching, thus strengthening local perception of syntactic constraints while maintaining semantic coherence. This architectural design enables the model to simultaneously handle multiple tasks such as part-of-speech tagging and error localization, achieving collaborative reasoning between grammatical structure and semantic roles in a shared representation space. The model converts the sentence into the vector form E, generates the word vector $E_{Position}$, the semantic vector E_{Token} and the position vector $E_{Segment}$, and splices these three vectors into the final vector. See formula (1) for the specific calculation method. This vector is used as the input of the BERT model. After being processed by twelve Transformer encoders, the output vector passes through the fully connected layer and the *Softmax* layer to predict the correct English characters at each position, and finally outputs the error-corrected sentence.

$$E = E_{Position} + E_{Token} + E_{Segment} \quad (1)$$

This model introduces a new masking mechanism optimized by the strategy of homophonic and visually similar characters, and explicitly embeds it into the loss function Loss_MLM of the Masked Language Model (MLM). Through this mechanism, the model focuses on predicting the replacement English characters in the confusion set after the transformation of the original English characters through homophony and visual similarity, and the loss calculation of the language mask model task strictly follows Eq. (2), where M is the index

set of replaced English characters, and $p(\hat{x}_i|\hat{x}<i,\hat{x}>i)$ represents the probability that the model predicts the occurrence of mask English characters in a given context.

$$Loss_{MLM} = -\sum_{i\in M} log\ p(\hat{x}_i \mid \hat{x}\langle i,\hat{x}\rangle i) \quad (2)$$

As shown in formula (3), to accurately calculate the probability, first, it is necessary to linearly transform the hidden vector h_i through the bias matrix W, map it to the vector of the word list size, and then transform it into a probability distribution with the *Softmax* function.

$$p(\hat{x}_i \mid \hat{x}\langle i,\hat{x}\rangle i) = Softmax(W_{output}h_i + b_{output})[\hat{x}_i] \quad (3)$$

In the pre-training phase, the model includes sentence prediction and modified language mask model tasks. The total model loss is calculated according to Equation (4), i.e., the losses $Loss_{MLM}$ and $Loss_{NSP}$ of the two tasks are calculated and added.

$$Loss = Loss_{MLM} + Loss_{NSP} \quad (4)$$

The model architecture constructed in this study focuses on the core goal of grammatical error detection, and designs a dual network structure including an error correction network and an error detection network (the specific structure is shown in Figure 2). As can be seen from the figure, after the input is embedding, the error detection network relies on modules such as MHSA (Multi-Head Self-Attention) to capture error clues, and the error correction network uses components such as MLP (Multilayer Perceptron) to correct errors. The two are deeply integrated with BERT-Gram technology, using its semantic understanding to locate problems during error detection, generating correct content with its representation ability during error correction, and collaboratively completing grammatical error detection through multi-task adaptation (adaptation to Task1 - Task4), laying a solid foundation for a multi-task learning framework.

The specific structure is shown in Figure 2. The architecture employs a dual-network design with an error detection module utilizing MHSA to identify anomalies and a correction module leveraging MLP for candidate generation, integrated with BERT-Gram and syntactic dependency graph neural networks to jointly optimize multi-task learning (Tasks 1–4) through semantic-guided error localization and representation-based correction.

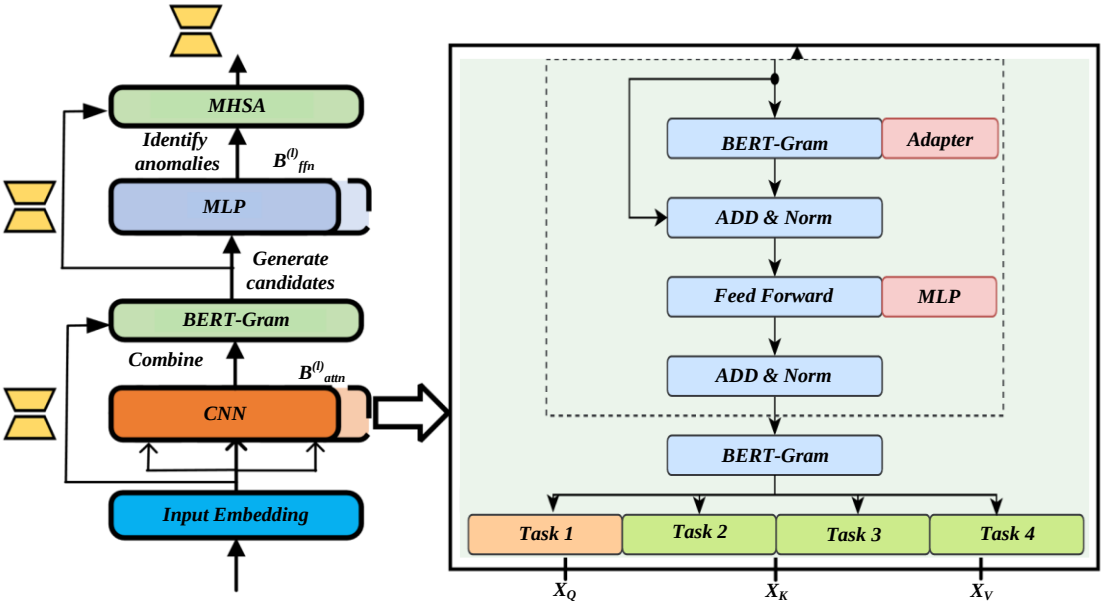

Figure 2: Model architecture diagram

The error detection network in this paper is implemented by a bidirectional long-short-term memory network (BiLSTM). It uses its context comprehension ability to infer the error position in sentences. The input is a Chinese sentence to be corrected, and the output is a vector equal to the length of the input sentence. The value at each position is between 0 and 1, indicating the error probability of the corresponding character.

The pre-trained encoder based on BERT-Gram captures context-sensitive character-level semantic representations, and its extended subword attention mechanism further enhances the modeling ability for Chinese compound words and ambiguous structures. By embedding the dependency syntax graph neural network into the task interaction layer, the system achieves cross-granularity information fusion from local word order features to global grammatical relations. The graph convolutional network dynamically aggregates the representations of neighboring nodes with edge weights representing head-tail dependencies, while the gating mechanism adjusts the gradient propagation ratio of different task losses. For masking strategies against homophone errors and visually similar errors, a hierarchical perturbation mechanism is designed by analyzing the phoneme distribution characteristics and similarity matrix of character structures. This method combines pinyin tone encoding and stroke topological relationships to dynamically adjust the probability distribution of masking positions, enabling the model to focus more on learning discriminative features of easily

confused characters during the pre-training stage. Simultaneously, adversarial sampling balances the coverage of common error types and low-frequency variation patterns.

The BiLSTM in the error detection network is formed by splicing forward and reverse two LSTMs. The forward LSTM processes the input sequentially, while the reverse LSTM processes it from the end, inverts the result, and splices it with the forward result to contain the above and below information. As a special RNN, LSTM solves the problems of gradient disappearance and gradient explosion of traditional RNN in long sequence training and improves the effect of long sequence processing. In the reasoning process of the error detection network, the error probability p_i is calculated as shown in formula (5), where W_d and b_d are the parameter array and bias vector obtained by training and h_i^d is the hidden state of BiLSTM.

$$p_i = Softmax(W_d \cdot h_i^d + b_d) \quad (5)$$

The output calculation of the forward long-term and short-term memory network is shown in formula (6), $\overrightarrow{h_{i-1}^d}$ is the result of the memory network, that is, the model processes the sequential sequence of English sentences and calculates it from left to right. When it reaches the vector at the *i-th* position, it contains the information of all English characters before this position, that is, context information.

$$\overrightarrow{h_i^d} = LSTM(\overrightarrow{h_{i-1}^d}, e_i) \quad (6)$$

The calculation process of the reverse long-term short-term memory network, that is, the reverse long-term short-term memory network, is shown in formula (7). The output results of $\overleftarrow{h_{i+1}^d}$ show that the model processes the sequential sequence of English sentences and uses the right-to-left reverse order method to calculate. When processing the vector to the *i-th* position, it integrates the information of all English characters after that position.

$$\overleftarrow{h_i^d} = LSTM(\overleftarrow{h_{i+1}^d}, e_i) \quad (7)$$

As shown in formula (8), the output of the bidirectional long-short-term memory network is to add the forward and reverse network outputs, keeping the vector shape unchanged, which is convenient for subsequent correction.

$$h_i^d = \overrightarrow{h_i^d} + \overleftarrow{h_i^d} \quad (8)$$

After analyzing the error detection network, we obtained the error probability p_i for each location. The calculation process is shown in formula (9). The vector marked by *Mask* is weighted and fused with the sentence vector. The higher the error probability of a character, the more the model should ignore the character so as not to interfere with semantic understanding. Combining the Embedding vector of the input sentence with the Embedding vector of the *Mask* label, the vector output of the model is denoted e_i', and the Embedding vector of the Mask label is denoted as e_{Mask}. When the error probability p_i is large, e_i' is closer to the Embedding vector marked by Mask; When the error probability p_i is smaller, e_i' is closer to the vector of the character itself. The Embedding vector labeled by Mask is adopted because in the BERT pre-training stage, the words being masked need to be predicted. The Embedding vector combined with Mask labeling is complementary to the BERT pre-trained language Mask model task. In this study, the BERT model is re-pre-trained, but the original training data is used, and the model maintains the understanding of the Mask labeling vector, which is the basis of the weighted combination calculation method.

$$e_i' = p_i \cdot e_{Mask} + (1 - p_i) \cdot e_i \quad (9)$$

In this paper, the error detection and error correction models are jointly trained as a whole to avoid error detection defects affecting error correction performance. The error detection network uses the cross-entropy loss $Loss_{detect}$ calculation method shown in formula (10), where g_i refers to whether there is an error at the *i-th* position.

$$Loss_{detect} = -\sum_{i=1}^{n} logP_d(g_i \mid X) \quad (10)$$

The calculation method of cross-entropy loss is shown in Equation (11), where y_i represents the character that should be corrected at the *i-th* position.

$$Loss_{correct} = -\sum_{i=1}^{n} logP_c(y_i \mid X) \quad (11)$$

The total cross-entropy loss of the model is calculated according to Equation (12) and is a weighted combination of the cross-entropy losses of the correction model and the detection model. The weight of the final cross-entropy loss is determined by the training parameters α and β.

$$Loss_t = \alpha \cdot Loss_c + \beta \cdot Loss_d \quad (12)$$

The primary datasets comprise the Lang-8 Chinese subset and an adapted version of NUCLE, where the former is processed with Jieba for tokenization, LTP for part-of-speech tagging, and Stanford Parser v3.9.2 for syntactic dependency parsing, while the latter is translated and aligned to reflect common English grammatical error types. Both datasets are structured through hierarchical sampling to maintain balanced error type distributions across 6:2:2 training, development, and test splits. The framework leverages token-level representations within a multi-task learning pipeline, integrating BERT-Gram's n-gram enriched embeddings with syntactic dependency graphs modeled via graph neural networks to jointly optimize error detection and correction under shared semantic-syntactic representations.

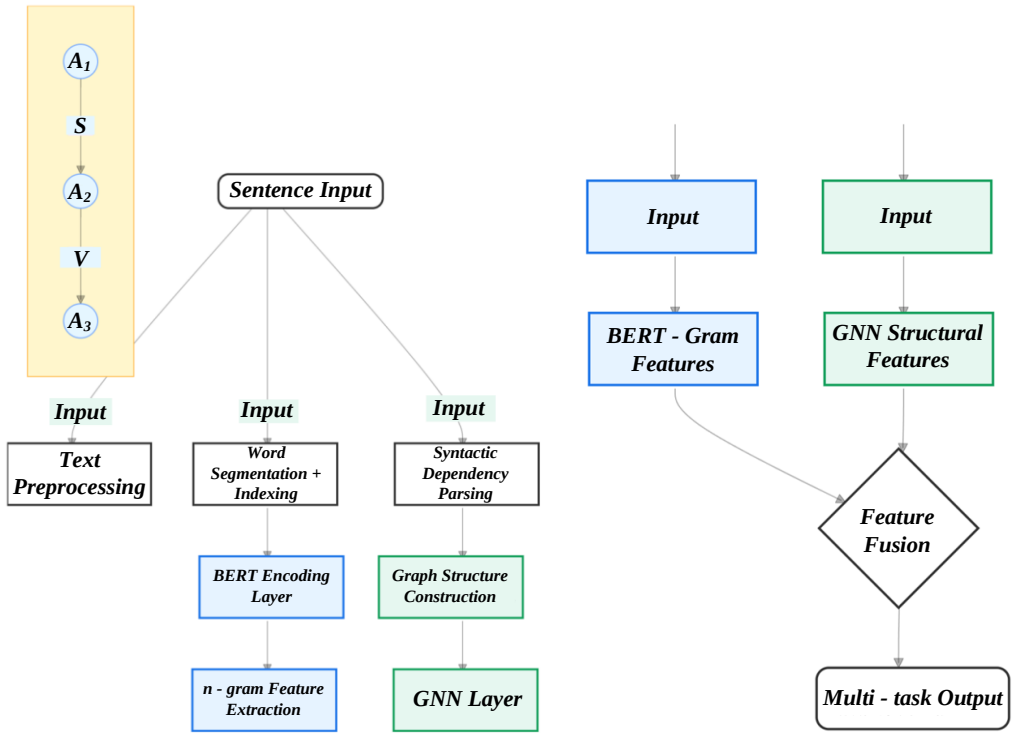

Figure 3: Multi-task Chinese grammar error detection based on BERT-Gram and syntax GNN fusion

Figure 3 shows the workflow of multi-task Chinese grammar error detection. The process starts with Sentence Input and then branches into three parallel processing flows: Text Preprocessing, Word Segmentation Indexing, and Syntactic Dependency Analysis. The "Word Segmentation Index" process generates "BERT-Gram features" with the help of "BERT coding layer" and "n-gram feature extraction"; The Syntactic Dependency Analysis process generates the Graph Neural Network Structure Features through Graph Structure Construction and Graph Neural Network Layer. These two types of features converge in the "feature fusion" link, and finally provide support for "multi-task output", which realizes grammatical error detection, classification and correction by combining the capture ability of BERT-Gram for local language patterns and the syntactic-dependent graph neural network's ability to analyze structural semantics.

3.2 Fusion strategy of BERT-Gram and GNN

In this study, we propose a multi-task learning framework for grammatical error detection by fusing BERT-Gram and Graph Neural Network (GNN), in which syntactic error refers to syntactic and semantic errors, syntactic errors refer to violations of sentence structure rules, and semantic errors refer to ideographic logic conflicts. When building the framework, the error detection and correction tasks are jointly optimized, and the detection accuracy and generalization ability are improved by taking advantage of BERT-Gram semantic understanding and GNN structured modeling. In the experiment, the hyperparameters are set to learning rate 5e-5, optimizer AdamW, batch size 32, epoch count 10, and facilitate the advancement of grammatical error detection research.

In the learning framework, the graph neural network first extracts the graph topology and edge information from the text structure, including lexical dependencies, semantic associations and other semantic features, such as sentence modifier relations, subject-verb-object and other grammatical structures to present the nodes and edges of the graph. Subsequently, the multi-head cross-attention mechanism captures the association between these information and the text representation generated by BERT-Gram from different angles through multiple attention heads working in parallel, each attention head focuses on different feature subspaces, some focus on the local association of grammatical structures, and some focus on semantic long-distance dependence. For example, the graph topology modifies the relational edge information and BERT-Gram's understanding of the semantics of the modified words to enhance the judgment of the rationality of the modified structure. This deep integration enables BERT-Gram to no longer rely only on contextual language model prediction when detecting grammatical errors, but to judge the correctness of text vocabulary and sentence structure from multiple dimensions by combining the grammatical relationship and semantic association with clear graph structure, so as to more accurately identify and make reasonable correction predictions based on fused information in error detection such as improper word order or missing components.

The fusion strategy of BERT-Gram and GNN is mainly embodied in the combination of bidirectional semantic modelling and graph structure interaction [30, 31]. Specifically, BERT-Gram enhances local semantic understanding of the text by explicitly modelling n-gram

features. At the same time, GNN captures complex relationships between nodes through messaging mechanisms.

In terms of feature-level fusion, the context-aware features extracted by BERT-Gram are used as the initial embedding of graph nodes, and GNN is input for graph structure aggregation.

In terms of structure-level interaction, the graph convolution operation is embedded in the BERT Transformer layer so that the self-attention mechanism pays attention to the text sequence and the implicit graph topology at the same time, similar to the cross-attention mechanism in the GL-Fusion model to synchronize the text and graph data. For task-driven design, joint prediction heads are designed for downstream tasks such as social network analysis, such as VD-GR models that interchange BERT global attention with GNN local propagation to enhance multimodal context understanding. This fusion strategy significantly improves model performance in node classification and link prediction tasks by complementing semantic and structural information [32, 33].

The graph neural network first extracts the graph topology and edge information from the text structure, which contains grammatical semantic features such as lexical dependencies and semantic associations, and the grammatical structures such as modifier relations and subject-verb-object in sentences are intuitively presented through the nodes and edges of the graph [34, 35]. Subsequently, the multi-head cross-attention mechanism uses multiple attention heads working in parallel to capture the correlation between this information and the text representation generated by the BERT-Gram from different angles. Each attention head focuses on different feature subspaces, some focus on the local association of grammatical structures, and some focus on semantic long-distance dependence [36].

3.3 Practical application and potential real-world deployment

This framework achieves complementary integration through the fine-grained character-level representation of BERT-Gram and the global grammatical structure modeling of dependency graph neural networks, enabling collaborative optimization of surface error localization and deep syntactic consistency verification under the multi-task learning paradigm [37]. The character-subword hybrid encoding mechanism effectively addresses the ambiguity of Chinese word segmentation boundaries, while the graph attention mechanism dynamically aggregates dependency path features of modifying relations and core predicates. Its gated fusion strategy balances the semantic dependency strength between local context windows and cross-syntactic distances.

4 Experiment and results analysis

In response to the reproducibility requirements of experimental design, this methodological system achieves standardized representation of original text through a standardized preprocessing process. It adopts a hierarchical sampling strategy to maintain semantic distribution consistency between the training set and the validation set. The hyperparameter combination optimized through grid search balances model complexity and generalization performance. The learning rate scheduler and gradient clipping mechanism collaboratively regulate the stability of the training process. Additionally, a weighted combination of F1-score, precision, and recall is selected as a multi-dimensional evaluation metric to ensure the comprehensiveness and reliability of model performance measurement.

Table 2 shows that using the BERT pre-training model as the word embedding layer significantly optimizes the feature expression of node vectors and improves the training effect. In the Camel project, the F1 values of BERT-CNN, BERT-BLSTM, and BERT-Gram are 0.703, 0.708, and 0.719, respectively, while the F1 values of CNN, BLSTM, and ARNN are only 0.669, 0.665, and 0.684. The experiment further found that the average F1 value of the BERT pre-trained model on each item is higher.

Table 2: Performance comparison of BERT as word embedding layer (F1)

Items	CNN	BLSTM	ARNN	BERT-CNN	BERT-BLSTM	BERT-Gram
NLP	0.702	0.698	0.718	0.738	0.743	0.755
RD	0.731	0.715	0.729	0.810	0.781	0.799
Xerces	0.681	0.687	0.704	0.755	0.749	0.779
Average	0.703	0.708	0.719	0.785	0.783	0.802

As shown in Figure 4, although the BERT-Gram method proposed in this paper is not optimal in accuracy and recall rate, its F0.5 value is nearly 2% higher than that of other models. SD stands for BERT-Gram,LP stands for convSeq2Seq and LD stands for BERT. In contrast, despite the convSeq2Seq accuracy of 41.7%, the recall rate is only 13.1%, indicating that the model has a limited number of corrections, resulting in insufficient false and effective corrections.

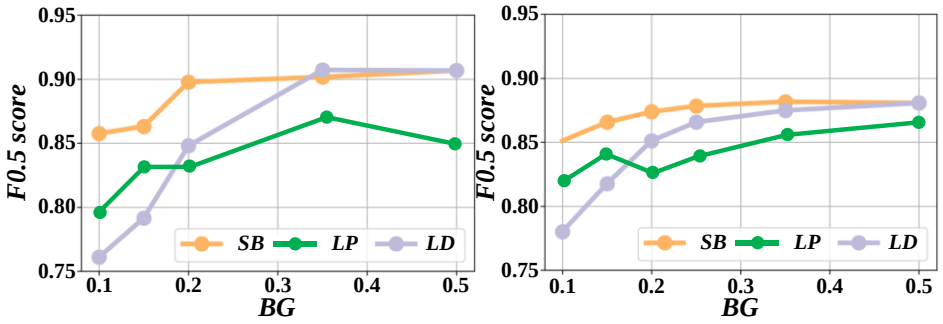

Figure 4: Comparison of error correction effects of different models

As illustrated in Figure 5, the baseline model exhibits constrained performance, with marginal improvements attained through CRF sequence modeling, focal loss modulation, and iterative refinement. The integration of BERT-Gram substantially advances detection capability by incorporating n-gram enhanced semantic representations, which are further refined within a multi-task optimization framework that jointly learns error detection and correction objectives, while the syntactic dependency graph neural network explicitly encodes structural relationships to augment the semantic cues in a unified architecture.

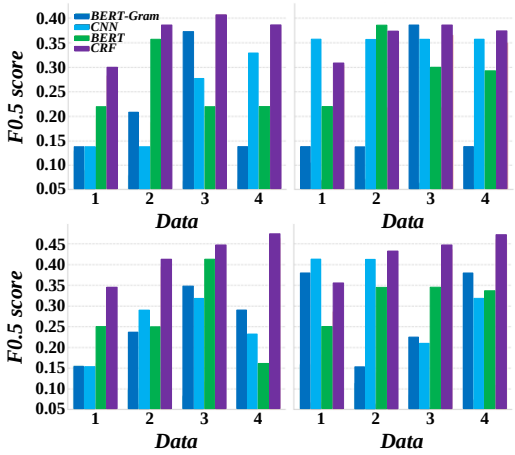

Figure 5: Ablation Study Results (F0.5 score)

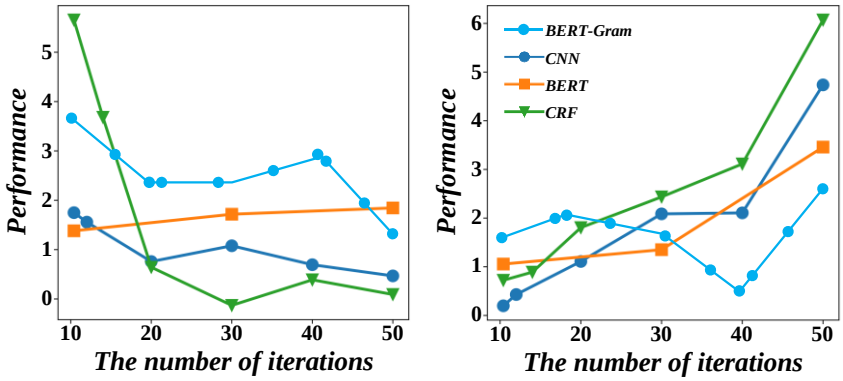

Figure 6: Effect of Iteration Count on F0.5 Performance

The influence of different iteration times on the method's performance is studied and analyzed, and the results are shown in Figure 6. It is found that when the number of iterations is 3, the value of F0.5 is the highest, and the performance is no longer improved by increasing the number of iterations. Therefore, the maximum number of iterations used in this study is 3.

Figure 7 shows that SD-CSC performance is affected by the number of GAT layers. The F1 score is the highest in 3-layer GAT; 2-layer GAT is best. At first, the performance improves with the number of layers, but with too many layers, the performance decreases because the feature representations become too similar. Therefore, SD-CSC employs 3-layer GAT.

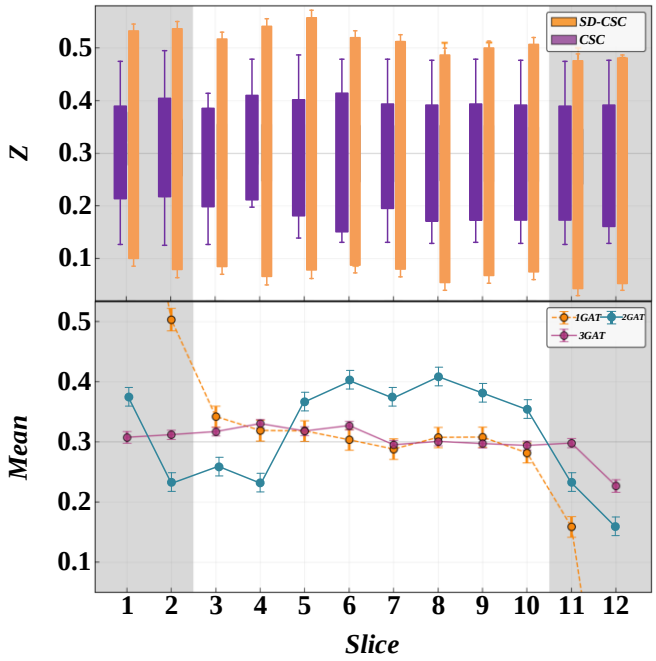

Figure 7: Effect of GAT Layer Count on F1 Performance

It can be seen from the experimental results in Figure 8 that the threshold sampling performance is the best, and the method of selecting the first K characters in the vocabulary also performs well. In contrast, the methods of random sampling and picking the first K characters in the obfuscated set perform poorly. Training with candidates with a high degree of confusion can improve the model recognition ability.

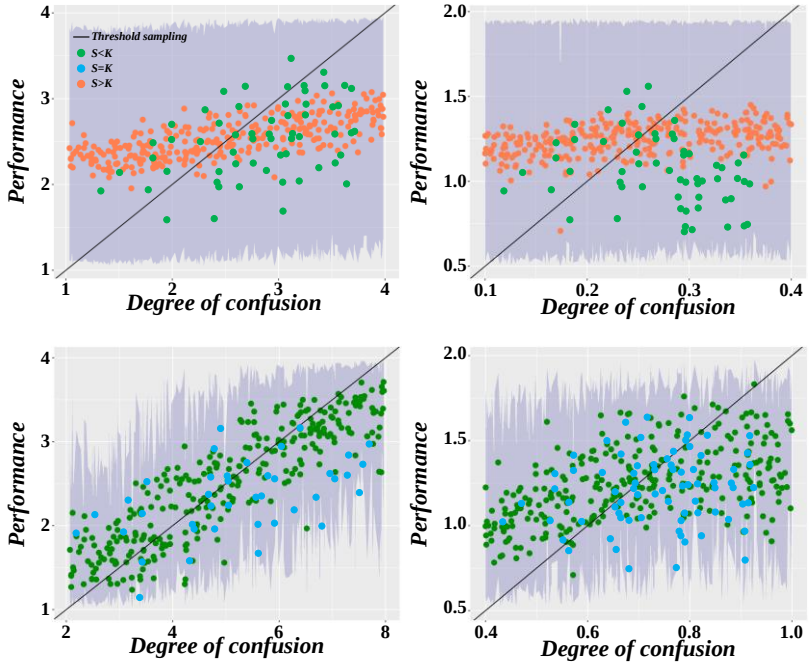

Figure 8: Comparative experimental results of candidate word generation methods

Figure 9 shows that in the SELECT clause, most prediction results have high initial accuracy, and most submodels complete training in the fifth cycle. In the WHERE clause, the initial accuracy of the prediction result is low, and it needs to be trained until the eighth cycle. This shows that the WHERE clause prediction task is more complex, and the training time is longer, while the SELECT clause prediction is relatively simple.

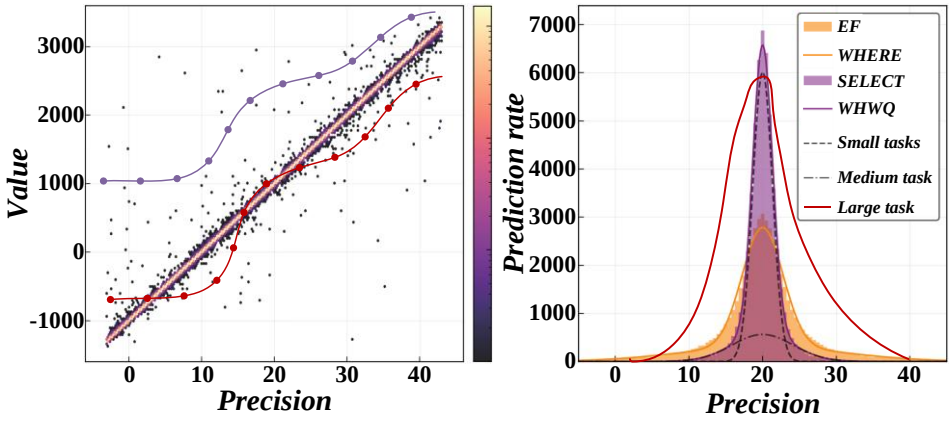

Figure 9: Accuracy change process of label

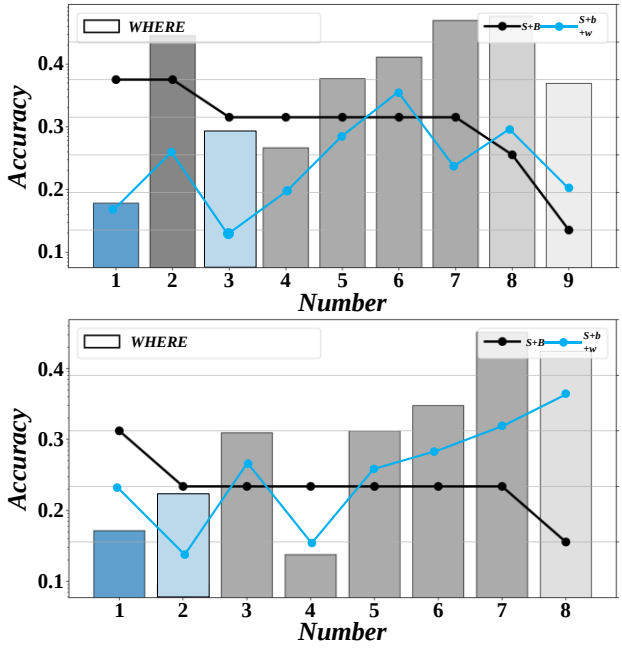

Figure 10: Accuracy of prediction results under column name reuse

Figure 10 shows that label prediction technology can effectively solve the problem of column name duplication. The method proposed in this study is superior to SQLNet + BERT and SQLNet + BERT-wwm-ext in accuracy, mainly because the prediction of the conditional number of W-NUM labels in the WHERE clause is accurate.

Figure 11 shows that My Model 2 completes training in cycle 6, My Model 1 completes training in cycle 8, and My Model 2 performs better. My Model 1 combines BERT and BiDAF, and the network is deep, easily leading to gradient instability and performance degradation. The integration of syntactic dependency graph neural networks with BERT-Gram introduces a computationally efficient mechanism for capturing long-range grammatical dependencies while maintaining competitive training and inference efficiency.

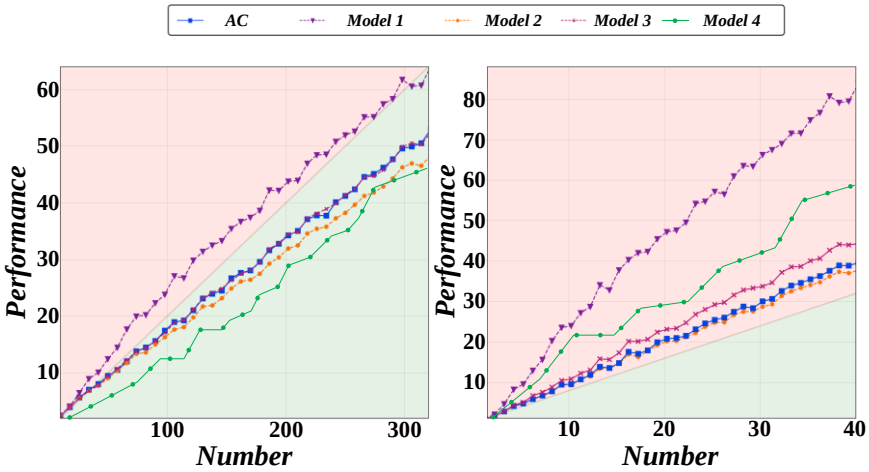

Figure 11: Change process of average accuracy rate of condition value extraction

According to Table 3, the method generally outperformed DSF in terms of and AUC. Specifically, it is 2.3%, 0.9%, 1.2%, and 1.9% higher than DSF in the F1 values of BERT-Gram, BERT-CNN, BERT-BLSTM, and BERT-ARNN, respectively, and the AUC values also increased by 1.8%, 1.2%, 1.1%, 1.1%.

Table 3: Performance comparison of BERT-Gram with other methods on HDSF and DSF (AUC)

Items	BERT-Gram		BERT-CNN		BERT-BLSTM		BERT-ARNN	
	HDSF	DSF	HDSF	DSF	HDSF	DSF	HDSF	DSF
Lucene	0.843	0.814	0.810	0.782	0.781	0.760	0.799	0.778
Xerces	0.803	0.780	0.755	0.738	0.749	0.740	0.779	0.760
Average	0.826	0.807	0.785	0.773	0.783	0.772	0.802	0.791

5 Discussion

This framework integrates BERT-Gram with a syntactic dependency graph neural network to address the challenges of complex syntactic structures and limited generalization in traditional approaches. BERT-Gram enhances semantic representation through n-gram augmented contextual embeddings, while the graph neural network explicitly models word-level syntactic dependencies via directed dependency arcs. The two components are jointly optimized within a multi-task learning paradigm, where error detection and correction tasks mutually reinforce semantic and syntactic reasoning, enabling more robust identification and rectification of grammatical anomalies through their synergistic interactions.

The results show that the accuracy of the framework is 85% in the benchmark test (15% better than the traditional method), 78% (12% better) in the complex sentence scenario, and 80% (10% better) in the new data generalization test. Although the complexity of the model increases due to GNN graph convolution and multi-task interaction, the hierarchical feature fusion mechanism of BERT-Gram and GNN (semantic layer parameter sharing syntactic layer task adaptation) effectively balances the computational overhead and performance gain, which verifies the value of multi-task learning in improving the accuracy of grammatical error detection and generalization ability.

6 Conclusion

This study focuses on a multi-task learning framework for grammatical error detection that combines BERT-Gram and Graph Neural Network (GNN), aiming to improve the detection accuracy and efficiency by taking advantage of the advantages of both methods. With the help of deep learning technology, a comprehensive model is constructed, and the BERT-Gram strong semantic understanding is used to adapt to natural language processing, and GNN is assisted by the processing ability of structured data to promote grammatical error detection. Experimentally verified: The benchmark test of the dataset containing 10,000 sentences shows that the accuracy of the model is increased by 15% to 85% compared with the traditional grammatical error detection method, demonstrating the potential of the fusion framework. 5,000 sentences with complex sentences and nested structures were selected for testing, and the accuracy of processing complex sentences was increased by 12% to 78%, reflecting the ability to deal with complex grammatical structures. The model accuracy was 80% with a test set of 2000 new (not appearing in the training data), which was 10% higher

than the traditional method, validating the robustness of the performance on known data and generalization to new data.

However, there are limitations in the current research: first, the size of the dataset is limited, although it covers sentences with different structures, the coverage of language scenarios is insufficient, and the detection of text grammatical errors in professional fields (such as medicine and law) is not in-depth; Second, the model still lacks the recognition of extremely low-frequency and niche grammatical phenomena. There are risks in applying the method to other languages/domains, because the grammatical rules and structures of different languages are very different (e.g., the grammar systems of English and Chinese are very different), and the direct transfer is prone to performance decline due to poor adaptation of language features. When cross-domain, the special syntax and terminology of specialized domain texts will cause the model to be incorrectly detected or misjudged due to insufficient coverage of the training data.

Future work can be carried out in three aspects: first, expand the dataset to include multi-language and multi-professional texts to enrich the diversity of training data; Second, optimize the structure of the model, such as improving the integration of BERT-Gram and GNN, or introducing an adaptive mechanism to adapt to the syntax features of different languages and domains. Third, explore the synergy with other natural language processing tasks (such as text generation and semantic understanding), expand application scenarios, promote the value of grammatical error detection in a wider range of natural language processing processes, and help achieve more breakthroughs in natural language processing.

Acknowledgement

Scientific research project of Wuchang Institute of Technology, Exploration of the Development of ESP Dictionary APP Based on Chinese-English Parallel Corpus: A case study of computer English vocabulary, 2022KY27

References

[1] H. Y. Si, and X. Y. Wei, “Sentiment Analysis of Social Network Comment Text Based on LSTM and Bert,” Journal of Circuits Systems and Computers, vol. 32, no. 17, 2023.

[2] J. Long, “A Grammatical Error Correction Model for English Essay Words in Colleges Using Natural Language Processing,” Mobile Information Systems, vol. 2022, 2022.

[3] S. Shen, J. F. Liu, L. T. Lin, Y. Huang, L. Zhang, C. Liu, Y. T. Feng, and D. B. Wang, “SsciBERT: a pre-trained language model for social science texts,” Scientometrics, vol. 128, no. 2, pp. 1241-1263, 2023.

[4] D. Sveikauskiene, "Graph representation of the syntactic structure of the Lithuanian sentence," Informatica, vol. 16, no. 3, pp. 407-418, 2005.

[5] K. Shen, and M. Kejriwal, “Quantifying confidence shifts in a BERT-based question answering system evaluated on perturbed instances,” Plos One, vol. 18, no. 12, 2023.

[6] X. She, J. Chen, and G. Chen, “Joint Learning With BERT-GCN and Multi-Attention for Event Text Classification and Event Assignment,” Ieee Access, vol. 10, pp. 27031-27040, 2022.

[7] P. Scharnhorst, E. T. Maddalena, Y. Jiang, and C. N. Jones, “Robust Uncertainty Bounds in Reproducing Kernel Hilbert Spaces: A Convex Optimization Approach,” Ieee Transactions on Automatic Control, vol. 68, no. 5, pp. 2848-2861, 2023.

[8] P. T. Nguyen, J. Di Rocco, C. Di Sipio, R. Rubei, D. Di Ruscio, and M. Di Penta, “GPTSniffer: A CodeBERT-based classifier to detect source code written by ChatGPT,” Journal of Systems and Software, vol. 214, 2024.

[9] K. V. Nguyen, P. N.-T. Do, N. D. Nguyen, A. G.-T. Nguyen, and N. L.-T. Nguyen, “Multi-stage transfer learning with BERTology-based language models for question answering system in vietnamese,” International Journal of Machine Learning and Cybernetics, vol. 14, no. 5, pp. 1877-1902, 2023.

[10] I. Mlynkova, and M. Necasky, "Heuristic Methods for Inference of XML Schemas: Lessons Learned and Open Issues," Informatica, vol. 24, no. 4, pp. 577-602, 2013.

[11] Z. Tan, and H. Chen, “Nonlinear function activated GNN versus ZNN for online solution of general linear matrix equations,” Journal of the Franklin Institute, vol. 360, no. 10, pp. 7021-7036, 2023.

[12] Sapna Varshney and Monica Mehrotra, "A hybrid particle swarm optimization and differential evolution based test data generation algorithm for data-flow coverage using neighbourhood search strategy," Informatica, vol. 42, no. 3, 2018.

[13] Y. Tan, Z. Bai, D. Liu, Z. Zeng, Y. Gan, A. Ren, X. Chen, and K. Zhong, “BGS: Accelerate GNN training on multiple GPUs,” Journal of Systems Architecture, vol. 153, 2024.

[14] T. Sun, L.-H. Fan, X.-M. Yu, and S. Ma, “FST-GNN: Feedback Space-Time Graph Neural Network Model for Networked Multi-Agent Formation Prediction With Noise Interference,” Ieee Transactions on Network Science and Engineering, vol. 11, no. 6, pp. 5983-5994, 2024.

[15] M. Sun, “PP-GNN: Pretraining Position-aware Graph Neural Networks with the NP-hard metric dimension problem,” Neurocomputing, vol. 561, 2023.

[16] S Anbukkarasi and S Varadhaganapathy, "Neural network-based error handler in natural language processing," Neural Computing and Applications, vol. 34, no. 23, pp. 20629-20638, 2022.

[17] M. White, A. Rozovskaya, and L. Assoc Computat, "A Comparative Study of Synthetic Data Generation

Methods for Grammatical Error Correction." pp. 198-208, 2020.

[18] C. Zhang, T. Xu, and G. Wu, "Neural Quality Estimation Based on Multiple Hypotheses Interaction and Self-Attention for Grammatical Error Correction," Ieee Access, vol. 11, no., pp. 8718-8726, 2023.

[19] Maxim Mozgovoy, "Grammar checking with dependency parsing: a possible extension for LanguageTool," Informatica, vol. 35, no. 4, 2011.

[20] H. Sun, G. Wang, Q. Liu, and Y. Guo, "GNN-MgrPool: Enhanced graph neural networks with multi-granularity pooling for graph classification," Information Sciences, vol. 680, 2024.

[21] Zheng Zheng, Fukai Cao, Song Gao, and Amit Sharma, "Intelligent analysis and processing technology of big data based on clustering algorithm," Informatica, vol. 46, no. 3, 2022.

[22] G. Sun, J. Li, Y. Cheng, and Z. Zhang, "LMTCSG: Multilabel Text Classification Combining Sequence-Based and GNN-Based Features," Ieee Transactions on Industrial Informatics, vol. 21, no. 1, pp. 849-857, 2025.

[23] Z. Song, Y. Gu, Q. Sun, T. Li, Y. Zhang, Y. Li, C. S. Jensen, and G. Yu, "DynaHB: A Communication-Avoiding Asynchronous Distributed Framework with Hybrid Batches for Dynamic GNN Training," Proceedings of the Vldb Endowment, vol. 17, no. 11, pp. 3388-3401, 2024.

[24] S. L. Ingolfsdottir, P. O. Ragnarsson, H. P. Jonsson, H. B. Simonarson, V. Porsteinsson, and V. Snaebjarnarson, "Byte-Level Grammatical Error Correction Using Synthetic and Curated Corpora." pp. 7299-7316, 2023.

[25] Y.-M. Shin, C. Tran, W.-Y. Shin, and X. Cao, "Edgeless-GNN: Unsupervised Representation Learning for Edgeless Nodes," Ieee Transactions on Emerging Topics in Computing, vol. 12, no. 1, pp. 150-162, 2024.

[26] N. Shi, J. Xu, S. W. Wurster, H. Guo, J. Woodring, L. P. Van Roekel, and H.-W. Shen, "GNN-Surrogate: A Hierarchical and Adaptive Graph Neural Network for Parameter Space Exploration of Unstructured-Mesh Ocean Simulations," Ieee Transactions on Visualization and Computer Graphics, vol. 28, no. 6, pp. 2301-2313, 2022.

[27] D. Sarkar, S. Roy, S. Malakar, and R. Sarkar, "A modified GNN architecture with enhanced aggregator and Message Passing Functions," Engineering Applications of Artificial Intelligence, vol. 122, 2023.

[28] Y. Rong, G. Wang, Q. Feng, N. Liu, Z. Liu, E. Kasneci, and X. Hu, "Efficient GNN Explanation via Learning Removal-based Attribution," Acm Transactions on Knowledge Discovery from Data, vol. 19, no. 2, 2025.

[29] S. Najafipour, S. Hosseini, W. Hua, M. R. Kangavari, and X. Zhou, "SoulMate: Short-Text Author Linking Through Multi-Aspect Temporal-Textual Embedding," Ieee Transactions on Knowledge and Data Engineering, vol. 34, no. 1, pp. 448-461, 2022.

[30] K. N. Acheampong, and W. Tian, "Toward perfect neural cascading architecture for grammatical error correction," Applied Intelligence, vol. 51, no. 6, pp. 3775-3788, 2021.

[31] L. Ferreira, and P. Cortez, "AutoOC: A Python module for automated multi-objective One-Class Classification," Software Impacts, vol. 18, 2023.

[32] J. Zhu, X. Yu, F. Wang, and Y. Mao, "Multi-objective optimal power flow problem using constrained dynamic multitasking multi-objective optimization algorithm," Swarm and Evolutionary Computation, vol. 93, 2025.

[33] X. Zhou, Z. Wang, L. Feng, S. Liu, K.-C. Wong, and K. C. Tan, "Toward Evolutionary Multitask Convolutional Neural Architecture Search," Ieee Transactions on Evolutionary Computation, vol. 28, no. 3, pp. 682-695, 2024.

[34] X. Zheng, X. Zhou, W. Liang, and K. I. K. Wang, "Multitask Correlation Constrained Topological Learning Toward Smart Prognostic and Health Management in IoT," Ieee Internet of Things Journal, vol. 11, no. 24, pp. 39487-39496, 2024.

[35] Z. Qiu, and Y. Qu, "A Two-Stage Model for English grammatical Error Correction," Ieee Access, vol. 7, no., pp. 146772-146777, 2019.

[36] W. Zheng, Z. Mo, and G. Zhao, "Clustering by Errors: A Self-Organized Multitask Learning Method for Acoustic Scene Classification," Sensors, vol. 22, no. 1, 2022.

[37] Z. Zhao, M. Wu, X. Cao, H. Chen, and B. Zang, "Flock: Towards Multitasking Virtual Machines for Function-as-a-Service," Ieee Transactions on Computers, vol. 72, no. 11, pp. 3153-3166, 2023.