Large Language Models for Graph Structure Learning

Anonymous ACL submission

Abstract

Graph Structure Learning (GSL) focuses on capturing intrinsic dependencies and interactions among nodes in graph-structured data by generating novel graph structures. Graph Neural Networks (GNNs) have emerged as promising GSL solutions, utilizing recursive message passing to encode node-wise interdependencies. However, many existing GSL methods heavily depend on explicit graph structural information as supervision signals, leaving them susceptible to challenges such as data noise and sparsity. In this work, we propose GraphEdit, an approach that leverages large language models (LLMs) to learn complex node relationships in graph-structured data. By enhancing the reasoning capabilities of LLMs through instruction-tuning over graph structures, we aim to overcome the limitations associated with explicit graph structural information and enhance the reliability of graph structure learning. Our approach not only effectively denoises noisy connections but also identifies node-wise dependencies from a global perspective, providing a comprehensive understanding of the graph structure. We conduct extensive experiments on multiple benchmark datasets to demonstrate the effectiveness and robustness of GraphEdit across various settings.

1 Introduction

Graph Structure Learning (GSL) is a burgeoning field of research that strives to unveil the underlying patterns and relationships within graph-structured data (Jin et al., 2020; Fatemi et al., 2021). In GSL, the primary focus lies in unraveling the latent relationships and dependencies that may not be immediately discernible from the raw data. By generating these novel graph structures, GSL empowers us to gain a more comprehensive understanding of the data, thereby facilitating various downstream tasks, such as node classification (Zhao et al., 2021a).

In recent years, graph neural networks (GNNs) have indeed captured significant attention and popularity due to their remarkable capacity to model and leverage relationships within graph-structured data (Garg et al., 2020; Buterez et al., 2022). GNNs excel in learning node-level representations by effectively aggregating and propagating information from neighboring nodes in a graph. This exceptional capability has sparked a revolution in the analysis of graph-structured data, enabling a more comprehensive understanding of the underlying node-wise connection patterns and interactions.

The ability to capture and leverage the intricate dependencies within graphs has undoubtedly propelled graph neural networks (GNNs) to the forefront of graph structure learning (Zhou et al., 2023). Notably, approaches like SLAPS (Fatemi et al., 2021), Nodeformer (Wu et al., 2022), and GT (Shi et al., 2021) incorporate neural networks that collaborate with GNNs to generate novel graph structures. These models undergo co-optimization, ensuring a seamless and integrated learning process. Moreover, recent studies such as SEGSL (Zou et al., 2023) and CoGSL (Liu et al., 2022a) have introduced dynamic methods for learning the graph structure. These approaches adaptively learn the graph structure based on predictions or representations generated by optimized GNNs.

While graph neural networks (GNNs) have demonstrated their high effectiveness, it is important to acknowledge that many of these approaches heavily depend on explicit graph structures, such as node links, as supervision signals for learning accurate representations. However, real-world graph domains often encounter challenges such as data noise and sparsity, which can compromise the reliability of these explicit graph structures.

To illustrate, let's consider a social network dataset where certain links are missing or incomplete due to privacy settings or limited data availability (Dai et al., 2022). Additionally, in recommender systems, the user-item interaction graph

may involve cold-start users or items, resulting in highly sparse links (Lin et al., 2021). Furthermore, various types of bias present in recommender systems introduce noise into the data (Wang et al., 2021b). In such cases, relying solely on explicit graph structures as supervision signals can lead to representations that are either inaccurate or biased. These challenges necessitate the development of more robust graph structure learning framework that can adapt to and overcome the impact of data imperfections in graph-structured data.

Contributions. In light of the challenges outlined earlier, this study seeks to explore how large language models (LLMs) can contribute to reasoning about the underlying graph structures. We introduce our proposed model, GraphEdit, which is designed to effectively refine graph structures. Our model's objective is twofold: first, to identify and address noisy connections between irrelevant nodes, and second, to uncover implicit node-wise dependencies. To achieve these goals, our model leverages the rich textual data associated with nodes in graph-structured data. By incorporating the text understanding ability of LLMs, specifically through the instruction-tuning paradigm, we enhance the understanding and representation of graph structures. This allows us to capture implicit dependencies among individual nodes that may not be explicitly encoded in the graph structure itself.

To thoroughly evaluate the performance of GraphEdit framework, we conducted extensive experiments, comparing it with state-of-the-art solutions. Additionally, we performed an in-depth ablation study and robustness analysis to validate the advantages and rationale behind our model. In order to ensure reproducibility, we have made our model implementation with source code available at the following anonymous link: `https://anonymous.4open.science/r/GraphEdit-41B3`.

2 Preliminaries

Graph-Structured Data. We define a graph using the tuple $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{T})$. Here, $\mathcal{V}$ represents a set of $N = |\mathcal{V}|$ nodes, $\mathcal{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix that captures the connections between nodes. Additionally, $t_n \in \mathcal{T}$ denotes the textual data associated with each node $n \in \mathcal{V}$ in graph $\mathcal{G}$, which consists of a sequence of L_n language tokens.

Graph Representation Learning. focuses on capturing meaningful and informative representations of nodes in a graph, enabling the analysis and modeling of intricate relationships and patterns within the graph data (Buterez et al., 2022). In recent years, Graph Neural Networks (GNNs) have emerged as promising approaches for capturing complex node-wise dependencies (Jin et al., 2020; Ji et al., 2019). By allowing nodes to exchange information with their neighbors, GNNs update their own representations and facilitate the propagation of information throughout the graph structure, enhancing our ability to understand and analyze the underlying graph data. However, in real-world graphs, noisy and missing connections are prevalent, and they significantly impair the performance of existing graph representation learning methods.

Problem Statement. Given the observed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{T})$ with noisy structural information, our objective is to improve the graph topology. This involves denoising the noisy connections within the graph data and uncovering the implicit relationships among nodes. By refining the original adjacency matrix $\mathcal{A}$ and obtaining a more informative graph structure $\tilde{\mathcal{A}}$, we can better capture the underlying node-wise dependencies, resulting in an updated graph $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{A}})$. This refinement process leads to a deeper understanding of the underlying graph structure, thereby improving the performance of downstream tasks by leveraging the updated graph structures $\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{A}})$.

3 Methodology

3.1 Instruction-Tuning LLM

Taking inspiration from the homophily property assumption discussed in studies such as (Gong et al., 2023; Li et al., 2023), it is suggested that nodes with similar attributes tend to have stronger connections. This concept has further evolved to explore the label consistency between nodes based on their connection patterns (Ma et al., 2021). Specifically, in the context of node classification tasks that rely on graph structures, the optimal situation involves maximizing connections within the same class while minimizing inter-class connections. Guided by this principle, our approach aims to leverage the knowledge of Large Language Models (LLMs) to reason about potential dependencies among nodes, taking into account the textual semantics associated with individual nodes.

During the prompt creation phase, we have meticulously defined two separate objectives within each prompt. The first objective is to evaluate the consistency of labels for the node pairs.

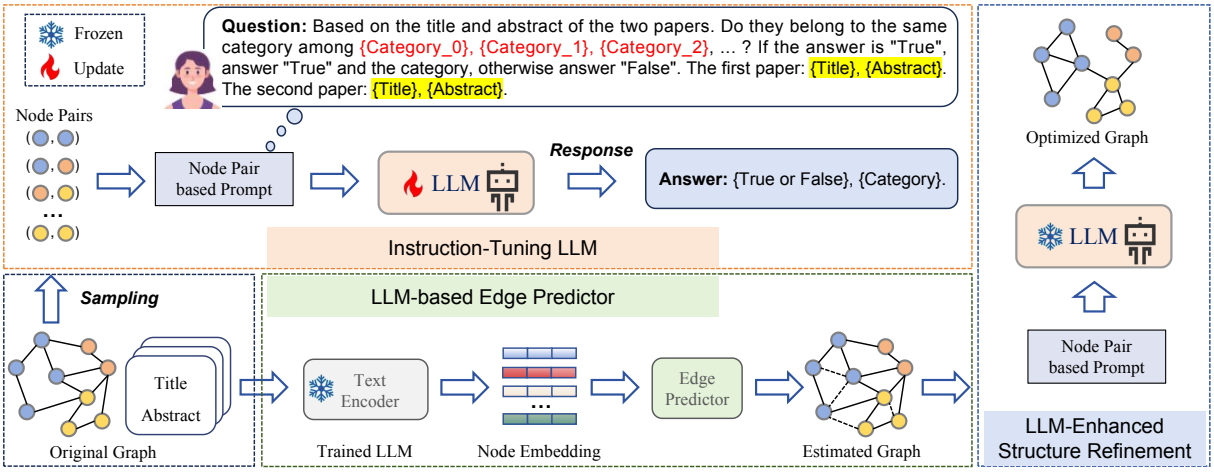

Figure 1: The model architecture of our proposed GraphEdit framework for graph structure learning.

Table 1: Prompt Instructions for Tuning LLMs.

Q: Based on the title and abstract of the two paper nodes. Do they belong to the same category among **{Category_0}**, **{Category_1}**, **{Category_2}**, ... ? If the answer is "True", answer "True" and the category, otherwise answer "False". The first paper: **{Title}**, **{Abstract}**. The second paper: **{Title}**, **{Abstract}**. A: **{True or False}**, **{Category}**.

This objective holds immense importance as it enables the language model to grasp the desired graph structures accurately. The second objective, which builds upon label consistency, involves determining the specific category to which these nodes belong. These carefully crafted prompts, encompassing these dual objectives, serve as valuable resources for instruction tuning of the language model.

In our methodology, we utilize a random sampling technique to select node pairs (n_i, n_j) from our training data N_{train}. These node pairs, used for tuning the LLMs, are randomly sampled from the training set N_{train}, where i and j represent two distinct nodes, and k represents any other node.

$$(n_i, n_j) \sim \text{Uniform}(N_{\text{train}} \times N_{\text{train}} - \{(n_k, n_k) | n_k \in N_{\text{train}}\}). \quad (1)$$

3.2 LLM-based Edge Predictor

To further enhance our analysis, we acknowledge the significance of identifying potential candidate edges in addition to the original graph structure. However, employing the trained language model (LLM) directly to traverse and reason over the entire graph presents a computational challenge, particularly for large graphs, due to the $\mathcal{O}(n^2)$ complexity, n represents the number of nodes in the graph. This computational complexity quickly becomes impractical as the graph size increases. To overcome this challenge, we propose the introduction of a lightweight edge predictor that aids the LLM in the selection process of candidate edges among the nodes in the graph $\mathcal{G}$.

In this approach, we leverage the node pairs that were previously sampled as the training set for the edge predictor. To ensure semantic consistency, we utilize the representations derived from the trained LLM for each node. This can be expressed as:

$$h_i = \text{LLM}(s_i), h_j = \text{LLM}(s_j), \quad (2)$$

We utilize the notation $< i, j >$ to represent a pair of nodes. The textual attributes associated with nodes i and j are denoted by s_i and s_j, respectively. The resulting representations, h_i and h_j, correspond to their respective nodes and retain the semantic knowledge and reasoning abilities transferred from the large language models.

After obtaining the node representations, we proceed to construct the training set labels y_e based on the node labels c_n using the following procedure:

$$y_e = \begin{cases} 1 & \text{if } c_i = c_j \\ 0 & \text{if } c_i \neq c_j \end{cases} \quad (3)$$

Next, we concatenate the representations of the two nodes in each node pair. We then feed the concatenated representation into a prediction layer denoted as $\eta(h_i || h_j)$, which allows us to obtain the probability of the edge's existence. We utilize cross-entropy as the loss function, denoted as $\mathcal{L}_{CE}(y_e, \hat{y}_e)$:

$$\hat{y}_e = \eta(h_i || h_j) \quad (4)$$

$$\mathcal{L}_{CE}(y_e, \hat{y}_e) = -[y_e log(\hat{y}_e + (1 - y_e) log(1 - \hat{y}_e)] \quad (5)$$

y_e denotes the ground truth label, and $\hat{y}_e$ represents the predicted probability of the edge's existence.

Table 2: Statistics of Experimental Datasets.

Dataset	# Nodes	# Edges	# Feat.	Classes
Cora	2708	5429	1433	7
Citeseer	3186	3703	4277	6
PubMed	19717	44335	500	3

3.3 LLM-enhanced Structure Refinement

To refine the graph structure, we employ the previously developed edge predictor to identify the top-k candidate edges for each node based on their estimated likelihood of existence. These candidate edges, along with the original edges of the graph, are then subjected to evaluation by the large language model (LLM) through a prompt, as depicted in Table 1. The LLM utilizes this information to determine which edges should be incorporated into the final graph structure. The graph structure refinement process can be summarized as follows:

$$\mathcal{A}' = \text{EdgePredictor}(\mathcal{H}_n) + \mathcal{A}, \tag{6}$$

$$\hat{\mathcal{A}} = \text{LLM}(\text{Prompt}(\mathcal{A}')) \tag{7}$$

The updated adjacency matrix, denoted as $\mathcal{A}'$, is obtained by combining the outputs of the edge predictor with the original adjacency matrix $\mathcal{A}$. This fusion process incorporates the edge predictor's predictions into the existing graph structure. Subsequently, the refined adjacency matrix $\hat{\mathcal{A}}$ is generated through the LLM's evaluation of the prompt applied to $\mathcal{A}'$. The LLM leverages its reasoning capabilities to make decisions regarding both the addition and deletion of edges in the final graph structure. Therefore, the refined adjacency matrix $\hat{\mathcal{A}}$ represents the LLM's informed choices, encompassing both the inclusion and exclusion of edges. This refined adjacency matrix serves as an input for downstream graph tasks, e.g., node classification.

In summary, our framework enhances the quality and structure of the final graph by incorporating the edge predictor's predictions and leveraging the reasoning capabilities of the LLM. This leads to the uncovering of implicit global node-wise dependencies and the denoising of noisy connections, resulting in an improved graph representation.

4 Evaluation

4.1 Experimental Settings

4.1.1 Datasets

To evaluate the performance of our GraphEdit method, we carefully selected three representative datasets: Cora, PubMed, and Citeseer. These datasets are widely recognized as benchmarks for graph learning tasks. In these datasets, each node represents a publication, and the edges represent citations between them. i) **Cora** dataset comprises papers classified into seven computer science domains: Case-Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, and Theory. It provides a diverse range of topics within the field of computer science. ii) **PubMed** dataset focuses on medical literature and categorizes papers into three distinct categories: Diabetes Mellitus, Experimental, Diabetes Mellitus Type 1, and Diabetes Mellitus Type 2. This dataset offers valuable insights into various aspects of diabetes research. iii) **Citeseer** dataset consists of academic papers from six different areas within computer and information science: Agents, Machine Learning (ML), Information Retrieval (IR), Databases (DB), Human-Computer Interaction (HCI), and Artificial Intelligence (AI).

4.1.2 Baselines

To comprehensively validate the effectiveness of our GraphEdit model, we compare it with 13 graph structure learning baselines, categorized into three groups based on their training strategies.

Pre-Training Models. GSR (Zhao et al., 2023), STABLE (Li et al., 2022), and SUBLIME (Liu et al., 2022b) are advanced pre-training models specifically designed to refine graph quality and enhance the effectiveness of graph representation learning. The training process involves two stages: first, enhancing the graph structure through pre-training, and then utilizing this refined structure to train GNNs for various downstream tasks.

Iter-Training Models. SEGSL (Zou et al., 2023), CoGSL (Liu et al., 2022a), and GEN (Wang et al., 2021a) employ the iterative training where two components are developed simultaneously. They adaptively learn the graph structure based on predictions or representations generated by an optimized GNN. The learned structure is then used to train a new GNN model in the subsequent iteration.

Co-Training Models. Notable examples of co-training models include Nodeformer (Wu et al., 2022), WSGNN (Lao et al., 2022), GT (Shi et al., 2021), SLAPS (Fatemi et al., 2021), Gaug (Zhao et al., 2021b), IDGL (Chen et al., 2020), and GRCN (Yu et al., 2020). In these models, the neural networks responsible for generating the graph structure are co-optimized alongside GNNs. This co-

Table 3: Accuracy comparison between GraphEdit and various state-of-the-art baselines. These refined structures are then fed into the downstream GCN encoder for representation learning in the node classification task.

Model	Cora	Citeseer	PubMed
GCN	87.36 ± 1.60	78.87 ± 2.18	87.37 ± 0.77
GRCN	84.13 ± 0.37	74.23 ± 1.18	85.20 ± 0.10
IDGL	88.63 ± 0.44	80.85 ± 0.07	88.30 ± 0.12
GAug	86.72 ± 0.63	77.61 ± 1.02	84.48 ± 0.37
GEN	86.53 ± 0.63	80.38 ± 0.72	87.04 ± 0.11
SLAPS	81.99 ± 1.57	73.17 ± 0.87	85.21 ± 0.18
GT	88.34 ± 0.35	78.46 ± 0.48	86.69 ± 0.19
CoGSL	82.07 ± 0.51	78.84 ± 0.11	OOM
WSGNN	89.59 ± 0.17	80.88 ± 0.48	87.17 ± 0.19
SUBLIME	85.04 ± 0.37	43.73 ± 7.08	86.03 ± 0.33
STABLE	88.75 ± 0.35	75.67 ± 0.98	86.30 ± 0.15
Nodeformer	88.56 ± 1.01	80.28 ± 0.57	87.93 ± 0.26
GSR	87.56 ± 1.19	78.77 ± 1.56	85.61 ± 0.55
SEGSL	87.49 ± 0.66	78.91 ± 0.52	87.57 ± 0.37
GraphEdit	$\mathbf{90.90 \pm 1.16}$	$\mathbf{81.85 \pm 1.42}$	$\mathbf{94.09 \pm 0.28}$

optimization ensures a more integrated and effective learning process, as both components mutually benefit from each other's improvements.

4.1.3 Implementation Details

In our GraphEdit model, we use Vicuna-v1.5 as our LLM, trained using the LoRA method. The model backbone consists of a two-layer GCN with a hidden size of 128. For our experiments, we divide the Cora, Citeseer, and PubMed datasets into three parts: training, validation, and testing. Following a ratio of 6:2:2, as mentioned in (He et al., 2023; Tang et al., 2023; Wen and Fang, 2023), ensures a consistent approach to dataset division. To train both the LLM and the Edge Predictor, we randomly sample 20,000 node pairs from the training set as training data. During the selection of candidate edges, we experiment with different top-k values ranging from 1 to 5. This exploration enables us to investigate the impact of varying the number of selected edges and determine the optimal setting. To ensure the robustness of our results, we repeat all experiments 10 times and calculate the mean and standard deviation of the outcomes. To facilitate fair comparisons, we tune the parameters of various baselines using a grid search strategy.

4.2 Performance Comparison

In our analysis of node classification tasks across three datasets, we compare our LLM-GSE model against various GSL baselines. The results are presented in Table 3, where "OOM" denotes out of memory error. Following existing GSL methods (Zhou et al., 2023; Wu et al., 2022), we utilize *accuracy* as the evaluation metric. From the comprehensive data analysis, we draw three key observations:

Obs 1: Remarkable Performance of GraphEdit. Our GraphEdit model demonstrates superior performance compared to existing graph structure learning methods across the three datasets, establishing itself as a state-of-the-art solution. The remarkable outcomes underscore the capacity of GraphEdit to enhance graph structures by unveiling implicit global dependencies and efficiently eliminating noisy connections among nodes in a graph. Through this process, GraphEdit not only improves the accuracy of graph structure learning but also enhances the overall quality and reliability of the learned graph representations.

Obs 2: Limitation of Existing GSL Approaches. Among the various GSL baselines, only a subset consistently outperforms the standard GCN, while some even impede the performance of downstream graph representation. These findings shed light on the limitations of alternative solutions that heavily rely on the original graph structures for supervision labels. However, it is crucial to acknowledge that the observed connections between nodes can often be noisy and incomplete, posing challenges for GSL methods in generating high-quality graph representations. In contrast, our GraphEdit capitalizes on the reasoning capabilities of LLMs to incorporate external semantics into the graph structure learning. By doing so, we enhance the overall quality of the learned representations in downstream tasks with our refined graph structures.

Obs 3: Performance Variation across Datasets. When analyzing the performance of GraphEdit, we observe a significant improvement on the PubMed dataset compared to Cora and Citeseer. Unlike Cora and Citeseer, PubMed has a larger number of nodes. Consequently, when training with an equal number of node pairs, the LLM encounters a more diverse range of situations in PubMed. Moreover, unlike Cora where missing abstracts are common, the textual information in PubMed's nodes is consistently abundant and detailed. Additionally, with only three categories, the PubMed dataset presents a less complex classification challenge. The same volume of sampling in PubMed enables the LLM to encounter a greater variety of edges associated with each category compared to the other two datasets.

Table 4: Model ablation study of our GraphEdit framework in terms of accuracy.

Model	**Cora**		**Citeseer**		**PubMed**	
GCN	87.36 ± 1.60		78.87 ± 2.18		87.37 ± 0.77	
MLP	77.32 ± 3.23		71.63 ± 1.47		85.83 ± 0.62	
w/o GNN	72.75		68.52		90.15	
Instruction	**-prompt**	**-prompt-w/o-ca**	**-prompt**	**-prompt-w/o-ca**	**-prompt**	**-prompt-w/o-ca**
GraphEdit w/o Add	88.38 ± 1.06	87.90 ± 1.89	80.03 ± 2.16	79.61 ± 2.20	91.83 ± 0.38	90.63 ± 0.51
GraphEdit w/o Del	90.52 ± 1.26	$\mathbf{89.96 \pm 1.25}$	81.43 ± 1.97	$\mathbf{80.19 \pm 2.11}$	89.63 ± 0.50	88.60 ± 0.60
GraphEdit	$\mathbf{90.90 \pm 1.16}$	89.52 ± 1.49	$\mathbf{81.85 \pm 1.42}$	79.84 ± 1.95	$\mathbf{94.09 \pm 0.28}$	$\mathbf{91.98 \pm 0.45}$

4.3 Model Ablation Study

To analyze the impact of different components on the performance of LLM-GSL, we conducted ablation experiments from two key perspectives. The experimental results are presented in Table 4.

Instruction-Tuning Paradigm. In Table 4, the "-prompt" notation refers to the utilization of a two-stage instruction-tuning paradigm (as shown in Table 1) to fine-tune the LLM. These instructions include the task of predicting both the existence of edges and the specific category of connected nodes. On the other hand, "-prompt-w/o-ca" indicates fine-tuning the LLM with simplified instructions that do not involve forecasting the specific node category.

Graph Structure Refinement. The "GraphEdit w/o Add" variant denotes that the GraphEdit is designed specifically for the deletion of edges from the original graph. This variant's primary objective is to identify and remove unnecessary or irrelevant edges, thus refining the graph structure. In contrast, the "GraphEdit w/o Del" variant refers to the functionality of the GraphEdit in appending candidate edges to the original graph structure. This approach aims to enrich the graph by introducing potentially valuable connections between nodes.

Based on the findings presented in Table 4, we can observe three significant phenomena:

• (i) These findings emphasize the importance of considering both edge existence and type prediction tasks within the fine-tuning process of GraphEdit. The exclusion of node type prediction in the "-prompt-w/o-ca" condition results in a loss of precision in node category matching.

• (ii) The performance of "GraphEdit w/o Add" sheds light on the impact of solely denoising the edges of the original graph, revealing limited effectiveness, particularly in datasets like Cora and Citeseer where the original edges are relatively sparse. However, the significant performance boost observed when adding candidate edges to the GCN highlights the ability of GraphEdit to capture implicit global dependencies among different nodes. These findings underscore the importance of leveraging both edge deletion and addition strategies, along with the reasoning capabilities of the LLM, to optimize the original graph structures.

• (iii) The performance difference between GraphEdit and "w/o GNN" emphasizes the need to incorporate GNN-encoded structural information for node classification tasks. "w/o GNN" solely relies on LLMs to infer node class based on GraphEdit's text understanding ability, without integrating the downstream GNN encoder to preserve graph topology. Therefore, including the GNN encoder is crucial for improved node classification performance as it captures the graph structure within the latent representation space.

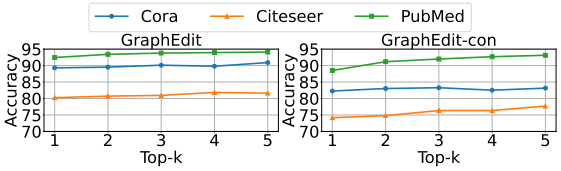

Figure 2: Impact study of edge candidate selection.

4.4 Impact of Edge Candidate Selection

To investigate the impact of different quantities of candidate edges on the model effectiveness, we analyze the performance across three datasets by varying the k-values from 1 to 5 (Figure 2). Generally, we observe that higher k-values tend to improve the model's performance. However, on the Cora and Citeseer data, the performance boost plateaus beyond $k = 3$, while on PubMed, it stabilizes around $k = 4$. This suggests that there is a threshold k-value beyond which the performance of GraphEdit stabilizes without significant further improvements. These findings provide insights for determining an optimal quantity of candidate edges, ensuring efficient utilization of computational resources while maintaining satisfactory performance.

Table 5: GraphEdit's performance in capturing inherent node relationships without explicit graph structure.

Method	Cora	Citeseer	PubMed
GCN	**87.36 ± 1.60**	**78.87 ± 2.18**	87.37 ± 0.77
MLP	77.32 ± 3.23	71.63 ± 1.47	85.83 ± 0.62
GraphEdit-con	83.27 ± 2.02	77.71 ± 2.17	**93.12 ± 0.29**

4.5 Graph Structure Construction

To further showcase the capability of our proposed GraphEdit framework in uncovering implicit node inter-dependencies, we assess its performance on the three datasets without the original graph structure. Results are shown in Table 5, where "GraphEdit-con" denotes the use of the graph structure constructed by GraphEdit alone. Remarkably, even in the absence of the original graph structure, GraphEdit delivers commendable performance. Notably, on the PubMed dataset, GraphEdit outperforms the original graph structure, highlighting its potential in text-rich scenarios. While GraphEdit falls short of surpassing the original structure on Citeseer, it achieves comparable results. Thus, this analysis confirms the effectiveness of our model in capturing inherent node relationships, even without an explicit graph structure.

Table 6: Performance on the PubMed with injected noisy edges at different rates (0.05, 0.1, 0.15, 0.2, 0.25).

Method	Attack Rate				
	0.05	0.1	0.15	0.2	0.25
GCN	86.06	85.13	84.28	83.61	83.26
IDGL	86.20	85.29	83.54	84.18	82.39
WSGNN	85.94	85.24	84.59	83.64	84.21
GraphEdit	**94.07**	**94.14**	**94.16**	**94.20**	**94.27**

4.6 Model Robustness Study against Noise

To investigate the noise resistance of GraphEdit, we injected varying proportions of noise (0.05 to 0.25) into the original graph structures of the three datasets. IDGL and WSGNN were selected as benchmarks and subjected to the same noisy conditions. Results are detailed in Table 6. The analysis reveals limited noise resistance in IDGL and WSGNN. In contrast, our GraphEdit method maintains stable performance. Surprisingly, on the PubMed dataset, increasing random noise edges actually improves GraphEdit's performance. This suggests effective noise edge elimination while retaining beneficial edges introduced as noise.

Table 7: Performance comparison with other LLMs.

Model	Cora	Citeseer
GCN	87.36 ± 1.60	78.87 ± 2.18
ChatGPT 3.5	85.30 ± 2.15	78.76 ± 2.19
ERNIE-Bot-turbo	86.99 ± 1.50	79.20 ± 2.25
Vicuna-7B	87.47 ± 1.22	79.55 ± 2.17
BLOOMZ-7B	84.87 ± 1.58	79.47 ± 2.28
Llama-2-7B	84.83 ± 1.94	78.65 ± 1.93
ChatGLM2-6B	80.92 ± 2.53	74.47 ± 2.09
AquilaChat-7B	86.31 ± 2.05	78.17 ± 2.42
GraphEdit	**88.38 ± 1.06**	**80.03 ± 2.16**

4.7 Comparison with other LLMs

We compared GraphEdit with commonly used LLMs (Brown et al., 2020; Sun et al., 2020; Chiang et al., 2023; Yong et al., 2022; Touvron et al., 2023; Du et al., 2021; BAAI, 2023) to evaluate their denoising capabilities on the original graph structures of the Cora and Citeseer datasets, using the same prompt shown in Table 1. The results are summarized in Table 7. GraphEdit outperforms other LLMs significantly in denoising on both datasets, demonstrating the effectiveness of our instruction tuning approach. Notably, ERNIE-Bot-turbo, Vicuna-7B, and BLOOMZ-7B perform well on the Citeseer dataset, although their performance is less impressive on the Cora dataset. This discrepancy can be attributed to the frequent occurrence of missing abstracts in Cora's nodes, which hampers the decision-making process of the LLMs and impacts the final graph structure.

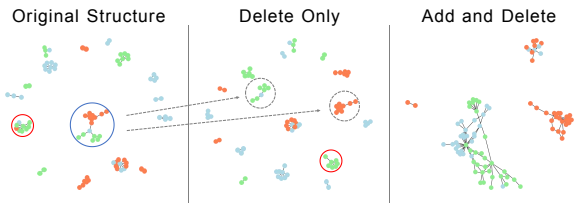

Figure 3: Visual analysis with random sampled 20 nodes and their 1-hot neighbors on the PubMed dataset.

4.8 Visual Analysis

In this section, we visually compare the original graph structure of PubMed with the optimized graph structure using Figure 3. The figures are arranged as follows: the original graph structure is on the left, the GraphEdit removed structure is in the middle, and the structure after adding and then removing edges is on the right. In the original graph structure, the central node faced classification challenges due to its neighboring nodes belonging to three different categories. However, GraphEdit

Table 8: Case study of GraphEdit on PubMed data

Query: Based on the title and abstract of the two papers. Do they belong to the same category among Diabetes Mellitus Type 1, Diabetes Mellitus Type 2, or Diabetes Mellitus, Experimental? If the answer is "True", answer "True" and the category, otherwise answer "False". The first paper: Title: {Node 2601 Title} Abstract: {Node 2601 Abstract}. The second paper: Title: {Node 6289 Title} Abstract: {Node 6289 Abstract}.
GT: True. Diabetes Mellitus Type 1.
The prediction results of our GraphEdit: True. Diabetes Mellitus, Experimental.

effectively addressed this issue by removing the neighbors of different categories around the central node, enabling accurate category determination.

Additionally, the original structure had a mixed area of three categories, which was successfully split into two distinct substructures after GraphEdit processing, simplifying the classification task. Moreover, the modified structure on the right maintained intra-class connections while eliminating inter-class links. These observations highlight the ability of GraphEdit not only to denoise the graph but also to restructure it in a way that greatly facilitates the task of node classification for the GCN.

4.9 Case Study

To demonstrate the advantages of predicting node consistency rather than directly predicting node categories, we present a clear example from the PubMed dataset. Table 8 showcases a straightforward case where nodes 2601 and 6289 are connected in the original PubMed graph structure and belong to the same category. During the inference process of GraphEdit, although it did not precisely predict the specific category of these two nodes, it successfully identified the consistency of their categories. This instance highlights how the training approach of GraphEdit effectively reduces the error rate in the LLM's inference, focusing on capturing the underlying consistency rather than precise categorization. This example serves to illustrate the benefits of prioritizing node consistency prediction, emphasizing the ability of the LLM-GSL approach to capture meaningful patterns and relationships in the graph structure, even if it falls short of precisely categorizing individual nodes.

5 Related Work

Graph Structure Learning. Various models have been developed to enhance our understanding and optimization of graph structures. Early works like Dropedge (Rong et al., 2019) and Neuralsparse (Zheng et al., 2020) focused on graph denoising through edge-dropping. LDS (Franceschi et al., 2019) modeled structures using Bernoulli distributions. More recent approaches, like IDGL (Chen et al., 2020) and GRCN (Yu et al., 2020), leverage node representations for structure formation. IDGL employs a weighted cosine function, while GRCN uses dual GNNs for structure derivation. WSGNN (Lao et al., 2022) employs variational inference for joint learning of node labels and graph structure. In contrast, SUBLIME (Liu et al., 2022b) explores unsupervised learning with a structure bootstrapping contrastive framework. However, these methods heavily rely on explicit node connections, making them susceptible to data noise and sparsity.

Large Language Models for Graphs. Recent research has begun exploring the application of Large Language Models (LLMs) in learning with graph-structured data. For instance, TAPE (He et al., 2023) utilizes LLMs to predict ranked classification lists for nodes, providing detailed explanations. KEA (Chen et al., 2023) enriches node text by incorporating knowledge entities. Additionally, ENG (Yu et al., 2023) leverages LLMs to generate new nodes, enhancing Graph Neural Network (GNN) performance in few-shot learning. GraphGPT (Tang et al., 2023) investigates the fusion of GNNs and LLMs, developing a customized LLM for graphs. However, none of these approaches address the challenges of noisy and incomplete data commonly found in graphs. In contrast, this work harnesses the reasoning capabilities of LLMs to robustly and effectively optimize the structure of graph-structured data.

6 Conclusion

We present a novel large language model called GraphEdit, which is designed for graph structure learning. Our model has the capability to identify noisy connections between nodes and uncover implicit relations between non-connected nodes, enabling the optimization of the graph structure. To achieve this, we integrate the power of LLMs with a lightweight edge predictor that we have developed. This integration allows our model to refine the graph structures, aligning them with the reasoning knowledge of LLMs. We conducted extensive experiments across various settings to evaluate the performance of our model. The results consistently demonstrate the superiority of GraphEdit. Additionally, through in-depth investigation, we further validate the rationale behind our model design.

7 Limitation

It is important to acknowledge that while the results in this study are promising, there are still limitations to be addressed in future studies.

Firstly, while our proposed GraphEdit has shown promising results, it is crucial to examine its performance across a wider range of graph structures. Future studies should explore different types of graph data, such as knowledge graphs, and biological networks, to assess the generalizability and adaptability of our model.

Secondly, real-world graph structures often undergo changes over time, making it essential to investigate how GraphEdit can handle dynamic and evolving graphs. Future research should explore strategies to adapt and update the model as new nodes, edges, or attributes are added or modified within the graph. This investigation will enable the model to stay up-to-date and maintain its effectiveness in dynamic environments.

Lastly, enhancing the interpretability and explainability of GraphEdit is an important avenue for further investigation. Developing techniques to extract meaningful insights and explanations from the model's reasoning process will help users better understand and trust the model's decisions.

References

BAAI. 2023. Aquilachat-7b.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *NeurIPS*, 33:1877–1901.

David Buterez, Jon Paul Janet, Steven J Kiddle, Dino Oglic, and Pietro Liò. 2022. Graph neural networks with adaptive readouts. *NeurIPS*, 35:19746–19758.

Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *NeurIPS*, 33:19314–19326.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. 2022. Towards robust graph neural networks for noisy graphs with sparse labels. In *WSDM*, pages 181–191.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*.

Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. 2021. Slaps: Self-supervision improves structure learning for graph neural networks. *NeurIPS*, 34:22667–22681.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *ICML*, pages 1972–1982. PMLR.

Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. 2020. Generalization and representational limits of graph neural networks. In *ICML*, pages 3419–3430. PMLR.

Shengbo Gong, Jiajun Zhou, Chenxuan Xie, and Qi Xuan. 2023. Neighborhood homophily-based graph convolutional network. In *CIKM*, pages 3908–3912.

Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. 2023. Explanations as features: Llm-based features for text-attributed graphs. *arXiv preprint arXiv:2305.19523*.

Tao Ji, Yuanbin Wu, and Man Lan. 2019. Graph-based dependency parsing with graph neural networks. In *ACL*, pages 2475–2485.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *KDD*, pages 66–74.

Danning Lao, Xinyu Yang, Qitian Wu, and Junchi Yan. 2022. Variational inference for training graph neural networks in low-data regime through joint structure-label estimation. In *KDD*, pages 824–834.

Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn. In *KDD*, pages 925–935.

Shouheng Li, Dongwoo Kim, and Qing Wang. 2023. Restructuring graph for higher homophily via adaptive spectral clustering. In *AAAI*, volume 37, pages 8622–8630.

Xixun Lin, Jia Wu, Chuan Zhou, Shirui Pan, Yanan Cao, and Bin Wang. 2021. Task-adaptive neural process for user cold-start recommendation. In *WWW*, pages 1306–1316.

Nian Liu, Xiao Wang, Lingfei Wu, Yu Chen, Xiaojie Guo, and Chuan Shi. 2022a. Compact graph structure learning via mutual information compression. In *WWW*, pages 1601–1610.

Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. 2022b. Towards unsupervised deep graph structure learning. In *WWW*, pages 1392–1403.

Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. 2021. Is homophily a necessity for graph neural networks? In *ICLR*.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2021. Masked label prediction: Unified message passing model for semi-supervised classification. In *IJCAI*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8968–8975.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2023. Graphgpt: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, and Xing Xie. 2021a. Graph structure estimation neural networks. In *WWW*, pages 342–353.

Wenjie Wang, Fuli Feng, Xiangnan He, Xiang Wang, and Tat-Seng Chua. 2021b. Deconfounded recommendation for alleviating bias amplification. In *KDD*, pages 1717–1725.

Zhihao Wen and Yuan Fang. 2023. Augmenting low-resource text classification with graph-grounded pre-training and prompting. *SIGIR*.

Qitian Wu, Wentao Zhao, Zenan Li, David P Wipf, and Junchi Yan. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. *NeurIPS*, 35:27387–27401.

Zheng-Xin Yong, Hailey Schoelkopf, Niklas Muennighoff, Alham Fikri Aji, David Ifeoluwa Adelani, Khalid Almubarak, M Saiful Bari, Lintang Sutawika, Jungo Kasai, Ahmed Baruwa, et al. 2022. Bloom+1: Adding language support to bloom for zero-shot prompting. *arXiv preprint arXiv:2212.09535*.

Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. 2020. Graph-revised convolutional network. In *ECML/PKDD*, pages 378–393.

Jianxiang Yu, Yuxiang Ren, Chenghua Gong, Jiaqi Tan, Xiang Li, and Xuecang Zhang. 2023. Empower text-attributed graphs learning with large language models (llms). *arXiv preprint arXiv:2310.09872*.

Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. 2021a. Heterogeneous graph structure learning for graph neural networks. In *AAAI*, volume 35, pages 4697–4705.

Jianan Zhao, Qianlong Wen, Mingxuan Ju, Chuxu Zhang, and Yanfang Ye. 2023. Self-supervised graph structure refinement for graph neural networks. In *WSDM*, pages 159–167.

Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2021b. Data augmentation for graph neural networks. In *AAAI*, volume 35, pages 11015–11023.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In *ICML*, pages 11458–11468. PMLR.

Zhiyao Zhou, Sheng Zhou, Bochao Mao, Xuanyi Zhou, Jiawei Chen, Qiaoyu Tan, Daochen Zha, Can Wang, Yan Feng, and Chun Chen. 2023. Opengsl: A comprehensive benchmark for graph structure learning. *arXiv preprint arXiv:2306.10280*.

Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, and Philip S Yu. 2023. Se-gsl: A general and effective graph structure learning framework through structural entropy optimization. In *WWW*, pages 499–510.