Large Language Models as Topological Structure Enhancers for Text-Attributed Graphs

Shengyin Sun[1⋆], Yuxiang Ren[2⋆], Jiehao Chen[3], and Chen Ma[1](✉)

[1]Department of Computer Science, City University of Hong Kong, Hong Kong, China
shengysun4-c@my.cityu.edu.hk, chenma@cityu.edu.hk
[2] Advance Computing and Storage Lab, Huawei Technologies, Shanghai, China
renyuxiang1@huawei.com
[3] China Academy of Industrial Internet, Beijing, China
chenjiehao@china-aii.com

Abstract. Inspired by the success of Large Language Models (LLMs) in natural language processing (NLP), recent works have begun investigating the potential of applying LLMs in graph learning. However, most existing work focuses on utilizing LLMs as node feature augmenters, leaving employing LLMs to enhance topological structures an understudied problem. In this paper, we are dedicated to leveraging LLMs' text comprehension capability to enhance the topological structure of text-attributed graphs (TAGs). First, we propose using LLMs to help remove/add edges in the TAG. Specifically, we first let the LLM output the semantic similarity between nodes through delicate prompt designs, and then perform edge deletion/addition based on the similarity. Second, we propose using pseudo-labels generated by the LLM to improve graph topology; we introduce pseudo-label propagation as a regularization to guide the graph neural network (GNN) in learning proper edge weights. Finally, we incorporate the two aforementioned graph topological refinements into the GNN training, theoretically justify the benefits of the proposed topology refinements, and perform extensive experiments on real-world datasets to demonstrate the effectiveness of the proposed methods. Code available at `https://github.com/sunshy-1/LLM4GraphTopology`.

1 Introduction

Graphs are ubiquitous in various scenarios such as molecular graphs in chemistry [26] and social networks [25]. In the real world, the text-attributed graph (TAG) is also representative, which integrates both graph topological structure and text, enabling them to contain richer semantics. To effectively learn graph-structured data, graph neural networks (GNNs) have emerged. They introduce a message-passing mechanism to aggregate features from neighboring nodes, allowing the simultaneous incorporation of graph topology and node features [6].

In the classic GNN pipeline, the text attributes of nodes are usually encoded as shallow embeddings, for example, by Word2Vec [20]. However, shallow embeddings have several limitations, such as the inability to effectively capture

⋆ Two authors contributed equally to this work sentence-level semantics [19] and inadequacy in handling the phenomenon of polysemy [23]. To overcome them, recent works have introduced text encoding methods powered by language models (LMs), which obtain contextualized textual embeddings by fine-tuning LMs (e.g., BERT [4]) on the specific task.

Although LMs have achieved satisfactory results in various tasks, researchers are still dedicated to further increasing the scale of model parameters to capture more complex linguistic patterns [12,31,17]. Recently, this vision has been realized by large language models (LLMs). State-of-the-art LLMs such as ChatGPT [1] and GPT4 [21], with hundreds of billions of parameters, have demonstrated remarkable capabilities across a wide range of NLP tasks [11,32]. However, the application of LLM's capabilities to graph-structured data remains relatively underexplored. Early attempts focused on using LLMs to enhance node features and did not consider how to utilize LLMs for perturbing the graph structure. In this paper, we investigate how to harness LLMs to refine the topological structure of TAGs. The key motivation is that *we find TAGs contain many irrational/unreliable edges that can potentially have an adverse effect on the message-passing process in GNN* (we show in Fig. 1 that the proportion of potentially unreliable edges in the Cora and Citeseer datasets exceeds 15%), thus degrading the quality of learned node representations. Specifically, unlike other network architectures such as CNNs and MLPs that individually encode each entity (node), GNNs encode nodes through aggregation/propagation of node features. This means that if two nodes are connected by unreliable edges, their final aggregated representations will inevitably contain some noise. Therefore, it is necessary to remove unreliable edges through graph topology refinement.

To mitigate the impact of unreliable edges on model performance, we resort to LLMs for assistance. First, we explore the potential of LLMs in performing edge deletion/addition on TAGs. Specifically, through careful prompt design, we let the LLM output the relatedness between two nodes based on their textual attributes. Based on the obtained semantic-level similarity, we perform edge deletion/addition on TAGs, where we manage to keep (or add) edges between nodes with higher similarity and delete (or not add) edges between nodes with lower similarity. Second, we investigate leveraging pseudo-labels generated by LLMs to improve the graph topology, where we propose to incorporate pseudo-labels generated by LLMs into GNN training. In particular, we introduce the propagation of pseudo-labels as a regularization to guide the model in learning proper edge weights that benefit the separation of different node classes (i.e., enabling nodes with a high probability of belonging to the same class to connect more strongly, while promoting a better separation of nodes with a high probability of belonging to different classes). Finally, we combine the two proposed topology refinement methods with the GNN training. Our contributions are as follows:

– Integrating LLMs with graph data is challenging, and most of the existing works only focus on using LLMs for feature enhancement. In this paper, we make the first attempt at employing LLMs to improve topological structure.
– Two LLM-based graph topology refinement approaches are proposed. One involves using carefully crafted prompt statements to make the LLM generate

semantic-level similarity between nodes. The generated similarity can be used to guide the addition of reliable edges and the removal of unreliable ones. The other involves using high-quality pseudo-labels generated by the LLM for pseudo-label propagation. The process of pseudo-label propagation is introduced as a regularization for the GNN training, leading the model to learn proper edge weights, thereby achieving the goal of improving graph topology. Moreover, we theoretically analyze the benefits of the proposed methods.

2 Related Work

Label Propagation. Traditional LPAs solely rely on labels and do not consider features [35]. To better utilize the information contained in node features, various techniques have been proposed for learning edge weights, including using kernel functions [16,37], minimizing reconstruction error [13], and imposing sparsity constraints [9]. Wang et al. proposed an approach using LPA itself as a regularization to help the model learn edge weights [30], which is different from the adaptive LPA mentioned above using a specific regularization. However, the initial label matrix used for label propagation in [30] is constructed solely using the information from labeled nodes while ignoring the information in unlabeled nodes. In this paper, we utilize LLMs to generate pseudo labels to assist label propagation, which enables the information in unlabeled nodes to be utilized, thereby guiding the model to learn more proper edge weights.

Applying LLMs on TAGs. LLMs [1,28] have demonstrated great potential in various NLP tasks. However, how to apply LLMs to graph-structured data, such as TAGs, remains a challenge [27,38,34,22]. Chen et al. investigate the potential of LLMs in node classification tasks [2]. He et al. propose a novel feature 'TAPE' augmented by the LLM [8], which incorporates the original attributes of nodes in the TAG, predictions made by the LLM on node categories, as well as explanations from the LLM for making such predictions. As shown before, existing methods primarily utilize LLMs to enhance the node features in graphs. However, approaches leveraging LLMs to aid the enhancement of graph topological structures remain relatively underexplored. In this paper, we focus on exploring LLMs' potential in enhancing graph topological structures.

3 Preliminary

3.1 Notations

Sets are denoted by calligraphic letters (e.g., $\mathcal{S}$). $|\cdot|$ denotes the number of elements in the set (e.g., $|\mathcal{S}|$). Matrices and vectors are denoted as bold upper case letters (e.g., $\mathbf{X}$) and bold lower case letters (e.g., $\mathbf{x}$), respectively. Superscript $(\cdot)^\top$ denotes transpose for matrices or vectors. $\mathbb{R}^{m\times n}$ represents a real matrix space of dimension $m \times n$. $\mathbf{H}[i,:]$ is the i-th row of $\mathbf{H}$. $\mathbf{H}(i,j)$ represents the (i,j)-th element in the matrix $\mathbf{H}$. $\|\mathbf{v}\|$ is the norm of $\mathbf{v}$.

3.2 Text-attributed Graph and Graph Neural Network

Define the TAG as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathbf{A}\}$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathcal{X}$ is a set of text attributes, and $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ represents an adjacency matrix. More specifically, for each node v_i in $\mathcal{V}$, it corresponds to a piece of text $x_i \in \mathcal{X}$. If there is an edge between v_i and v_j, then $\mathbf{A}_{ij} = 1$; otherwise, $\mathbf{A}_{ij} = 0$. Under the node classification setting, v_i also corresponds to a label y_i that indicates which category the text attribute of node v_i belongs to. Mathematically, the representation of node u at the l-*th* GNN layer $\mathbf{h}_u^l$ can be expressed as: $\mathbf{h}_u^l = f^l(\boldsymbol{\Phi}; \mathcal{X}, \mathbf{A}) = \textsc{update}\left(\mathbf{h}_u^{l-1}, \mathrm{AGG}\left(\left\{\mathbf{h}_v^{l-1}, v \in \mathcal{N}_u\right\}\right)\right)$, where $\mathcal{N}_u$ is the set of neighbors of node u, $\textsc{Agg}(\cdot)$ is the operation of aggregating messages, $\textsc{update}(\cdot)$ is the operation of updating node u's representation by using its previous representation and aggregated messages. For the sake of simplicity, the $\textsc{Agg}(\cdot)$ and $\textsc{update}(\cdot)$ operation in the l-*th* layer can be further abstracted into a parameterized function $f^l(\cdot)$ with the learnable parameters $\boldsymbol{\Phi}$.

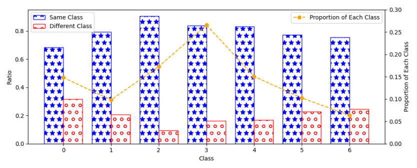
4 Methodology

4.1 Necessity of Graph Topology Enhancement

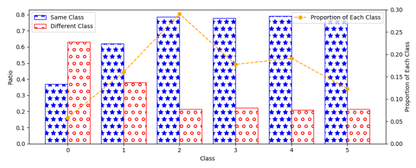
• **The topology enhancement is neglected by existing LLM-GNN works**. A straightforward approach to combining LLMs and GNNs is to utilize the text generation capability of LLMs to enrich the input of GNNs. Employing LLMs as attribute enhancers is exemplified in works such as: (i) TAPE [8]—LLM is used to generate pseudo-labels for each node and is required to provide explanations for generating such pseudo-labels. These pseudo-labels and explanations are concatenated with the original text attributes as input for the GNN; (ii) KEA [2]—LLM is employed to extract technical terms from the original text attributes and is tasked with elaborating on these terms in detail. The generated technical terms with descriptions, along with the original text attributes, are then encoded by the GNN. However, these works overlook a crucial point that distinguishes the GNN from other architectures (such as the CNN): GNNs encode nodes through the aggregation/propagation of node features, rather than encoding nodes individually. *In other words, it is challenging to ensure the quality of the final representation (after aggregation) of a node when it is connected to noisy nodes through unreliable edges, no matter how its own features are enhanced.* Therefore, we argue that the LLM-based topology enhancement is as important as the LLM-based attribute enhancement.

• **Observations from public datasets**. We investigate the proportion of potentially unreliable edges[4] in public datasets, as shown in Fig. 1. Overall, the proportion of potentially unreliable edges cannot be ignored (averaging over 15%). Taking Citeseer as an example, among edges where at least one endpoint

[4] A potentially unreliable edge refers to an edge connecting two endpoints belonging to distinct classes (assuming that we know the true labels of all nodes).

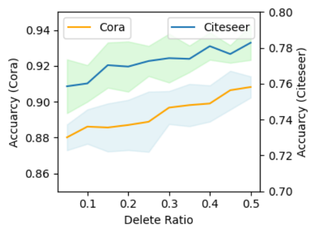


(a) Cora Dataset

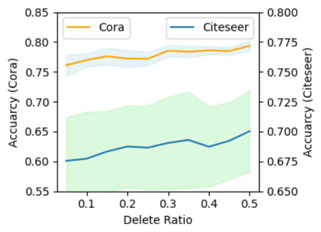


(b) Citeseer Dataset

Fig. 1: The proportion of edge endpoints with the same/different classes.



(a) General Setting



(b) Few-shot Setting

Fig. 2: The impact of removing potentially unreliable edges on accuracy.

belongs to the class "0", the proportion of potentially unreliable edges exceeds 60%. This suggests that nodes of class "0" exhibit a high probability of connecting to unrelated nodes, which, in turn, can lead to blurred boundaries between the representations of nodes with class "0" and those of nodes with other classes after feature aggregation. To further demonstrate the potential of topology enhancement, we explore the impact of removing these potentially unreliable edges. As shown in Fig. 2, the accuracy steadily improves with the increasing delete ratio, which aligns with our previous analysis.

4.2 LLM-based Edge Deletion and Addition

LLMs have introduced a novel paradigm called "*pre-train, prompt, and predict*". Under this paradigm, the LLM first undergoes pre-training in large-scale text datasets to learn general knowledge. Then, instead of fine-tuning the pre-trained model for each downstream task, the pre-trained model is prompted with a human-readable prompt according to the specific downstream task. The model generates the output based on the prompt and given input, without any task-specific fine-tuning of the model parameters.

From the perspective of the LLM's input/output, the edge deletion/addition shares the same form of prompt paradigm. Specifically, we need to input the texts of the two endpoints (corresponding to a certain edge) into the LLM and guide the LLM to output its judgment on deleting or adding the edge, i.e.,

$$r_{ij} = \mathrm{LLM}(\mathrm{Prompt}(x_i, x_j)), \qquad x_i, x_j \in \mathcal{X}, \tag{1}$$

where x_i and x_j are text attributes corresponding to the endpoints, $\mathrm{Prompt}(\cdot)$ is the designed prompt format, and r_g is the response text generated by the LLM. The well-designed prompt structure $\mathrm{Prompt}(\cdot)$ for the edge deletion/addition is:

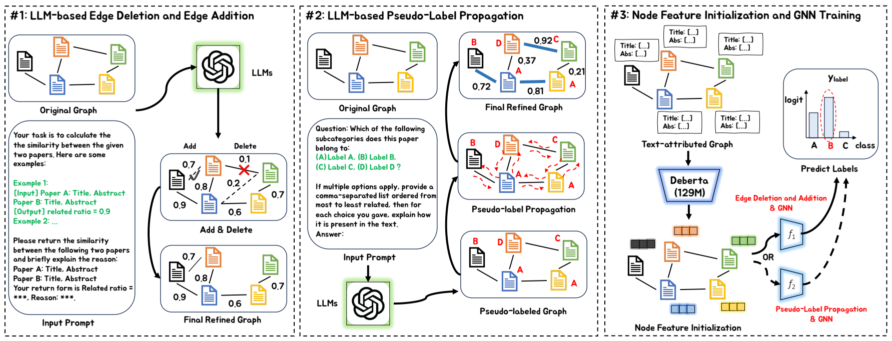

#1: LLM-based Edge Deletion and Edge Addition
Original Graph
LLMs
Your task is to calculate the similarity between the given two papers. Here are some examples:
Example 1:
[Input] Paper A: Title, Abstract
Paper B: Title, Abstract
[Output] related ratio = 0.9
Example 2: ...
Please return the similarity between the following two papers and briefly explain the reason:
Paper A: Title, Abstract
Paper B: Title, Abstract
Your return form is Related ratio = ***, Reason: ***.
Input Prompt
Add
Delete
Add & Delete
Final Refined Graph
#2: LLM-based Pseudo-Label Propagation
Original Graph
Question: Which of the following subcategories does this paper belong to:
(A) Label A. (B) Label B.
(C) Label C. (D) Label D ?
If multiple options apply, provide a comma-separated list ordered from most to least related, then for each choice you gave, explain how it is present in the text.
Answer:
Input Prompt
LLMs
Final Refined Graph
Pseudo-label Propagation
Pseudo-labeled Graph
#3: Node Feature Initialization and GNN Training
Title: [...]
Abs: [...]
Text-attributed Graph
Deberta (129M)
Node Feature Initialization
Edge Deletion and Addition & GNN
OR
Pseudo-Label Propagation & GNN
f_1
f_2
y_{label}
logit
A B C class
Predict Labels

Fig. 3: The overview of the proposed methods. (#1) LLM-based edge deletion/addition. (#2) LLM-based pseudo-label propagation. (#3) Integration into GNN training.

Your task is to calculate the similarity between the given two papers. Here are some examples where the input is the information for paper A and paper B, and the output is the estimated related ratio between paper A and paper B: \n
Example 1: **[Input]** **Paper A**: Title: [paper title ...] Abstract: [paper abstract ...] **Paper B**: Title: [paper title ...] Abstract: [paper abstract ...] **[Output]** Estimated Related Ratio between Paper A and Paper B: 0.x\n
Example 2: ...\n
Please return the similarity between the following two papers and briefly explain the reason: **Paper A:** Title: [paper title ...] Abstract: [paper abstract ...] **Paper B:** Title: [paper title ...] Abstract: [paper abstract ...]. Your return form is: Related ratio = ***, Reason: ***.

In the aforementioned prompt structure, we first describe the task to the LLM, that is, to estimate the similarity between two papers based on their titles and abstracts. To help the LLM better understand the task and prevent it from generating meaningless numbers, we then provide multiple examples as references. Finally, we include the text attributes related to the two endpoints (i.e., paper A and paper B), and request the LLM to output its estimations. After obtaining the similarity estimations given by the LLM, we can use a threshold-based method to determine which edges need to be added or removed, i.e.,

$$\mathbf{A}_{\mathrm{A-D}}(i,j) = \begin{cases} 1, & \text{if Extract}(r_{ij}) > \xi \\ 0, & \text{if Extract}(r_{ij}) \leq \xi \end{cases} \tag{2}$$

where $\mathbf{A}_{\mathrm{A-D}}$ is the adjacency matrix of the refined graph, Extract$(\cdot)$ is a function that extracts similarity numbers from LLM's response, ξ is a preset threshold.

4.3 LLM-based Pseudo-label Propagation

In this section, we first prompt the LLM to give pseudo-labels based on the text attributes of nodes. For convenience, we take the Citeseer dataset as an example to describe. The prompt structure on the Citeseer dataset is as follows:

> The title and abstract of the paper are as follows: Title: [paper title ...] Abstract: [paper abstract ...]\n
> **Question**: Which of the following subcategories does this paper belong to: (A) Agents, (B) ...? If multiple options apply, provide a comma-separated list ordered from most to least related, then for each choice you gave, explain how it is present in the text.\n
> **Answer**:

After obtaining the response text from the LLM, we choose the category that the LLM thinks the node v_i is most likely to belong to (the category ranked first in the response text) as the pseudo-label y_i^p. Inspired by the work [30], we hope to utilize the pseudo-labels generated by the LLM to optimize edge weights (in a label propagation manner) so that important edges can be highlighted in the message passing of GNNs. Formally, the LLM-based pseudo-label propagation containing K iterations can be written as:

$$\begin{cases} \mathbf{Y}^{(0)} = \left[\mathbf{y}_1^{(0)}, \ldots, \mathbf{y}_{|\mathcal{V}|}^{(0)}\right]^\top, \\ \mathbf{Y}^{(k)} = \bar{\mathbf{A}}\mathbf{Y}^{(k-1)} = \left[\mathbf{y}_1^{(k)}, \ldots, \mathbf{y}_{|\mathcal{V}|}^{(k)}\right]^\top, \quad k = 1, \cdots, K, \end{cases} \tag{3}$$

where $\mathbf{Y}^{(0)}$ is the initial label matrix for the label propagation, $\mathbf{y}_i^{(0)}$ for $i = 1, \ldots, |\mathcal{V}|$ is the one-hot label indicator vector generated according to the y_i^p, $\bar{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix (consistent with the normalization method in the GCN [14]) with the learnable adjacency matrix $\mathbf{A}$ and the diagonal degree matrix $\mathbf{D}$ for $\mathbf{A}$. The optimal edge weights can be obtained by minimizing the loss of predicted labels by the LLM-based pseudo-label propagation, i.e.,

$$\mathbf{A}^* = \arg\min_{\mathbf{A}} \frac{1}{|\mathcal{V}_{\text{train}}|} \sum_{v_i \in \mathcal{V}_{\text{train}}} \mathcal{L}_{\text{CE}}(\mathbf{y}_i{}^{\text{llm-lpa}}, \mathbf{y}_i), \tag{4}$$

where $\mathbf{A}^*$ is the optimized graph topology, $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss, $\mathcal{V}_{\text{train}}$ is the training set containing labeled nodes, $\mathbf{y}_i{}^{\text{llm-lpa}}$ is the predicted label distribution of node v_i after K iterations of propagation (i.e, $\mathbf{y}_i{}^{\text{llm-lpa}} = \mathbf{y}_i^{(K)}$), and $\mathbf{y}_i$ is the true one-hot label of node v_i. It should be noted that Eq. (4) merely demonstrates how to utilize the pseudo-labels from LLMs to help optimize the graph topology. In the next section, we will elaborate in more detail on how to integrate the LLM-based pseudo-label propagation into GNN training and train the whole model in an end-to-end manner.

4.4 LM Fine-tuning and GNN Training

We fine-tune the pre-trained LM DeBERTa-base [7] to encode the text attribute x_i of the node v_i into the vectorial feature, i.e.,

$$\mathbf{h}_i = \mathrm{LM}(x_i), \quad v_i \in \mathcal{V}, x_i \in \mathcal{X}, \tag{5}$$

where $\mathbf{h}_i$ is the text embedding for the node v_i, $\mathrm{LM}(\cdot)$ represents the fine-tuned LM model DeBERTa-base. To extract the most informative textual features tailored for the downstream task, we fine-tune the LM [8]. More specifically, we apply a multi-layer perceptron (MLP) to the output of the LM and minimize the cross-entropy loss between the LM's predictions and the labels for each node.

For the method of LLM-based edge deletion and edge addition, we can perform the GNN training by directly utilizing the adjacency matrix $\mathbf{A}_{\mathrm{A-D}}$ (see Section 4.2 for details). Formally, the process of message passing and model optimization can be written as follows:

$$\begin{cases} \mathbf{H}^{(k)} = \sigma\left(\bar{\mathbf{A}}_{\mathrm{A-D}}\mathbf{H}^{(k-1)}\mathbf{W}^{(k-1)}\right), \quad k = 1, \cdots, K, \\ \mathbf{W}^{*} = \underset{\mathbf{W}}{\arg\min} \frac{1}{|\mathcal{V}_{\mathrm{train}}|} \sum_{v_i \in \mathcal{V}_{\mathrm{train}}} \mathcal{L}_{\mathrm{CE}}(\mathbf{y}_i{}^{\mathrm{gcn}}, \mathbf{y}_i), \end{cases} \tag{6}$$

where $\sigma(\cdot)$ is an activation function, $\mathbf{W}^{(k)}$ is a learnable weight matrix in the k-*th* layer, $\bar{\mathbf{A}}_{\mathrm{A-D}}$ is the normalized adjacency matrix for $\mathbf{A}_{\mathrm{A-D}}$, $\mathbf{H}^{(k)}$ is the output matrix of the k-*th* layer (the initialization of $\mathbf{H}$ is based on the Eq. (5), i.e., $\mathbf{H}^{(0)} = \left[\mathbf{h}_1, \cdots, \mathbf{h}_{|\mathcal{V}|}\right]^{\top}$), $\hat{\mathbf{y}}_i{}^{\mathrm{gcn}}$ represents the predicted label distribution of node v_i by using GCN in Eq. (6), $\mathbf{W}$ denotes learnable parameters in the whole model, and $\mathbf{W}^{*}$ is the optimized parameters.

For the LLM-based pseudo-label propagation, the process of model optimization and topology optimization can be written as:

$$\begin{aligned} \mathbf{W}^{*}, \mathbf{A}^{*} = & \underset{\mathbf{W},\mathbf{A}}{\arg\min} \frac{1}{|\mathcal{V}_{\mathrm{train}}|} \sum_{v_i \in \mathcal{V}_{\mathrm{train}}} \Big[\mathcal{L}_{\mathrm{CE}}(\mathbf{y}_i{}^{\mathrm{gcn}}, \mathbf{y}_i) \\ & + \lambda \mathcal{L}_{\mathrm{CE}}(\mathbf{y}_i{}^{\mathrm{lpa}}, \mathbf{y}_i) + \beta \mathcal{L}_{\mathrm{CE}}(\mathbf{y}_i{}^{\mathrm{llm\text{-}lpa}}, \mathbf{y}_i)\Big], \end{aligned} \tag{7}$$

where λ and β are hyper-parameters, $\mathbf{y}_i^{\mathrm{lpa}}$ and $\mathbf{y}_i^{\mathrm{llm-lpa}}$ are two label distributions obtained from two different label propagation methods. Specifically, $\mathbf{y}_i^{\mathrm{llm-lpa}}$ is the label distribution obtained from the initial label matrix constructed using the pseudo-labels generated by the LLM (see Eq. (3) for details), while $\mathbf{y}_i^{\mathrm{lpa}}$ is the label distribution obtained from the initial label matrix constructed using the true labels from the training set (with the one-hot encoded labels for nodes outside the training set initialized as default values) [30]. The overall framework is provided in Fig. 3.

4.5 Theoretical Justification

Essentially, the proposed approaches aim to refine the message-passing process by applying hard/soft masks to graphs. In this section, we harness the shrinking theory [15,30,33] to understand the benefits of proposed methods in refining the message passing process. Define the embedding variation [36] on graphs as:

$$\mathcal{M}\left(\mathbf{H}^{(k)}\right)=\frac{1}{2}\sum_{i=1}^{|\mathcal{V}|}\sum_{j=1}^{|\mathcal{V}|}\left\|\frac{\mathbf{A}(i,j)}{\mathbf{D}(i,i)}\mathbf{h}_i^{(k)}-\frac{\mathbf{A}(j,i)}{\mathbf{D}(j,j)}\mathbf{h}_j^{(k)}\right\|^2, \tag{8}$$

where $\mathbf{A},\mathbf{D},\mathbf{H}^{(k)}$ are the previously refined adjacency matrix, diagonal degree matrix, and embedding matrix, respectively. For the convenience, we rewrite $\mathcal{M}\left(\mathbf{H}^{(k)}\right)$ as $\sum_{i=1}^{|\mathcal{V}|}\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)$. Utilizing the Taylor's theorem, we have

$$\begin{aligned}
\sum_{i=1}^{|\mathcal{V}|}\mathcal{M}_i\left(\mathbf{h}_i^{(k)}-\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right)&=\sum_{i=1}^{|\mathcal{V}|}\Big\{\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)-\\
&\quad\left[\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right]^\top\int_0^1\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}-\theta\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right)\mathrm{d}\theta\Big\}\\
&=\sum_{i=1}^{|\mathcal{V}|}\Big\{\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)-\left[\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right]^\top\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)-\left[\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right]^\top\\
&\quad\times\int_0^1\left[\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}-\theta\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right)-\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right]\mathrm{d}\theta\Big\}\\
&\le\sum_{i=1}^{|\mathcal{V}|}\Big\{\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)-\left[\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right]^\top\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)+\\
&\quad\int_0^1\left\|\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right\|\left\|\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}-\theta\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right)-\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right\|\mathrm{d}\theta\Big\}.
\end{aligned}\tag{9}$$

Note that

$$\mathbf{h}_i^{(k)}-\frac{1}{2}\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)=\mathbf{h}_i^{(k)}-\sum_{j=1}^{N}\left[\frac{\mathbf{A}(i,j)}{\mathbf{D}(i,i)}\mathbf{h}_i^{(k)}-\frac{\mathbf{A}(i,j)}{\mathbf{D}(j,j)}\mathbf{h}_j^{(k)}\right]=\mathbf{h}_i^{(k+1)}, \tag{10}$$

the Eq. (9) can be further expressed as:

$$\begin{aligned}
\sum_{i=1}^{|\mathcal{V}|}\mathcal{M}_i\left(\mathbf{h}_i^{(k)}-\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right)&\le\sum_{i=1}^{|\mathcal{V}|}\Bigg\{\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)-\left[\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right]^\top\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\\
&\quad+\int_0^1\left\|\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right\|\left\|2\theta\eta\left(1-\frac{\mathbf{A}(i,i)}{\mathbf{D}(i,i)}\right)\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right\|\mathrm{d}\theta\Bigg\}\\
&\le\sum_{i=1}^{|\mathcal{V}|}\left[\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)-\left(\eta-\eta^2\right)\left\|\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right\|^2\right].
\end{aligned}\tag{11}$$

The Eq. (11) indicates that when the value of η changes between 0 and 1, the inequality $\sum_{i=1}^{|\mathcal{V}|}\mathcal{M}_i\left(\mathbf{h}_i^{(k)}-\eta\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right)\le\sum_{i=1}^{|\mathcal{V}|}\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)$ always holds. We have shown in Eq. (10) that $\mathbf{h}_i^{(k+1)} = \mathbf{h}_i^{(k)} - \frac{1}{2}\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)$, so Eq. (11) at $\eta = \frac{1}{2}$ can be written as:

$$\sum_{i=1}^{|\mathcal{V}|}\mathcal{M}_i\left(\mathbf{h}_i^{(k+1)}\right) = \sum_{i=1}^{|\mathcal{V}|}\mathcal{M}_i\left(\mathbf{h}_i^{(k)} - \frac{1}{2}\nabla\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right)\right) \leq \sum_{i=1}^{|\mathcal{V}|}\mathcal{M}_i\left(\mathbf{h}_i^{(k)}\right). \tag{12}$$

This implies that the variation of node embeddings decreases (shrinking) as the message passing proceeds. In other words, embeddings of nodes shrink into distinct clusters in the space based on the graph topology. The motivation of the proposed methods is to eliminate unreliable connections in the graph topology, allowing related nodes (e.g., nodes of the same class) to be connected as much as possible. This enables the embeddings to shrink into distinct clusters according to their classes, thus clarifying the classification boundaries and enhancing the model's performance (we will verify this in subsequent experiments).

5 Experiments

5.1 Datasets and baselines

Experiments are conducted on four real-world TAGs: Cora [18], Citeseer [5], Pubmed [24], Arxiv-2023 [8]. We evaluate our methods by comparing them against 13 baselines from the following three categories: (1) Classical GNNs: GCN [14], GAT [29], GraphSAGE [6]; (2) LM-based methods: BERT [4], DeBERTa [7], GIANT [3], and variants combining classical GNN with DeBERTa encoding (GCN$_{\mathrm{DeBERTa}}$, GAT$_{\mathrm{DeBERTa}}$, GCN-LPA$_{\mathrm{DeBERTa}}$); (3) LLM-based methods: GPT-3.5-Turbo [1], TAPE [8] (the SOTA paradigm combining LLMs and GNNs, including TAPE$_{\mathrm{GAT}}$, TAPE$_{\mathrm{GCN}}$, and TAPE$_{\mathrm{GCN-LPA}}$).

5.2 Implementation Details

We adopt the GCN as the underlying graph learning architecture to validate the effect of LLM-based topology perturbations. The hyperparameters used for the GCN are consistent with previous studies [8,10]. As our goal is to investigate the effects of topology modifications, the hyperparameters associated with the GCN structure are not tuned specifically for each dataset. Considering the expenses associated with querying LLMs' APIs and the rate limit imposed by OpenAI, we randomly select 1,000 edges as candidate edges for edge deletion/addition in each dataset. For edge deletion, we choose candidate edges from the existing edges in each dataset. For edge addition, we select candidate edges based on the second-order neighbors of the nodes. Specifically, if two nodes are second-order neighbors and there is no existing edge between them, these two nodes will be considered as candidate node pairs for edge addition. The threshold ξ used in the deletion/addition process is tuned between 0.1 and 0.9 with an interval of 0.1. The threshold ξ is tuned between 0.1 and 0.9 (interval 0.1). The λ and β are tuned between 0 and 5 (interval 0.1). For the general setting, 60% of the nodes

Table 1: Experimental results in the general setting (**Best**, second Best).

Dataset	Cora	Pubmed	Citeseer	Arxiv-2023
GPT-3.5-Turbo	0.6769 ± 0.0000	0.9342 ± 0.0000	0.5929 ± 0.0000	0.7356 ± 0.0000
GAT	0.8573 ± 0.0065	0.8328 ± 0.0012	0.7423 ± 0.0178	0.6784 ± 0.0023
GCN	0.8690 ± 0.0151	0.8890 ± 0.0032	0.7298 ± 0.0132	0.6760 ± 0.0028
GraphSAGE	0.8573 ± 0.0065	0.8685 ± 0.0011	0.7361 ± 0.0190	0.6906 ± 0.0024
GIANT	0.8423 ± 0.0053	0.8419 ± 0.0050	0.7238 ± 0.0083	0.5672 ± 0.0061
BERT	0.7400 ± 0.0175	0.9058 ± 0.0046	0.7317 ± 0.0175	0.6840 ± 0.0122
DeBERTa	0.7385 ± 0.0127	0.9020 ± 0.0057	0.7313 ± 0.0194	0.6789 ± 0.0185
GAT$_{\text{DeBERTa}}$	0.8750 ± 0.0084	0.9312 ± 0.0083	0.7547 ± 0.0231	0.7704 ± 0.0043
GCN$_{\text{DeBERTa}}$	0.8778 ± 0.0137	0.9314 ± 0.0039	0.7508 ± 0.0066	0.7694 ± 0.0022
GCN-LPA$_{\text{DeBERTa}}$	0.8750 ± 0.0209	0.9446 ± 0.0030	0.7559 ± 0.0104	0.7831 ± 0.0038
GCN + LLM-based A-D	0.8815 ± 0.0180	0.9309 ± 0.0050	0.7578 ± 0.0079	0.7708 ± 0.0009
GCN + LLM-based LPA	**0.8828 ± 0.0191**	0.9469 ± 0.0037	0.7614 ± 0.0149	0.7853 ± 0.0022
GCN + LLM-based A-D & LPA	0.8778 ± 0.0183	**0.9475 ± 0.0036**	**0.7633 ± 0.0117**	**0.7858 ± 0.0027**
TAPE$_{\text{GAT}}$	0.8838 ± 0.0088	0.9316 ± 0.0115	0.7610 ± 0.0107	0.7807 ± 0.0023
TAPE$_{\text{GCN}}$	0.8833 ± 0.0046	0.9319 ± 0.0037	0.7571 ± 0.0046	0.7819 ± 0.0027
TAPE$_{\text{GCN-LPA}}$	0.8824 ± 0.0051	0.9461 ± 0.0023	0.7653 ± 0.0073	0.8020 ± 0.0029
TAPE$_{\text{GCN}}$ + LLM-based A-D	0.8907 ± 0.0138	0.9329 ± 0.0045	0.7625 ± 0.0101	0.7840 ± 0.0012
TAPE$_{\text{GCN}}$ + LLM-based LPA	0.8875 ± 0.0151	**0.9475 ± 0.0049**	**0.7692 ± 0.0059**	**0.8033 ± 0.0037**
TAPE$_{\text{GCN}}$ + LLM-based A-D & LPA	**0.8916 ± 0.0096**	0.9472 ± 0.0056	0.7684 ± 0.0076	0.8032 ± 0.0023

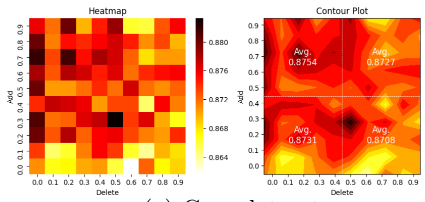
Table 2: Experimental results in the few-shot setting (**Best**, second Best).

Dataset	Cora	Pubmed	Citeseer	Arxiv-2023
GAT$_{\text{DeBERTa}}$	0.7272 ± 0.0315	0.6462 ± 0.0685	0.6703 ± 0.0429	0.4185 ± 0.1070
GCN$_{\text{DeBERTa}}$	0.7552 ± 0.0094	0.6462 ± 0.0664	0.6647 ± 0.0337	0.4960 ± 0.0458
GCN-LPA$_{\text{DeBERTa}}$	0.7588 ± 0.0132	0.6382 ± 0.0550	0.6657 ± 0.0319	0.5050 ± 0.0437
GCN + LLM-based LPA	**0.7630 ± 0.0134**	**0.6535 ± 0.0427**	**0.6780 ± 0.0341**	**0.5175 ± 0.0392**
TAPE$_{\text{GAT}}$	0.7990 ± 0.0054	0.8573 ± 0.0107	0.7223 ± 0.0240	0.6747 ± 0.0826
TAPE$_{\text{GCN}}$	0.8190 ± 0.0050	0.8668 ± 0.0147	0.7200 ± 0.0168	0.7365 ± 0.0271
TAPE$_{\text{GCN-LPA}}$	0.8232 ± 0.0033	0.8815 ± 0.0195	0.7198 ± 0.0084	0.7382 ± 0.0278
TAPE$_{\text{GCN}}$ + LLM-based LPA	**0.8275 ± 0.0066**	**0.8930 ± 0.0256**	**0.7307 ± 0.0143**	**0.7428 ± 0.0272**

are designated for the training set, 20% for the validation set, and the remaining 20% are set aside for the test set. For the few-shot setting, we randomly select 20 nodes from each class to form the training set, 500 random nodes for the validation set, and 1,000 random nodes from the remaining pool for the test set.

5.3 Performance Comparisons

General setting. Table 1 shows the experimental results under the general setting, where "LLM-based A-D" represents "LLM-based Edge Deletion and Edge Addition", "LLM-based LPA" represents "LLM-based Pseudo-label Propagation", and "LLM-based A-D & LPA" represents the combination of "LLM-based A-D" and "LLM-based LPA". It can be seen that the LLM-based topological structure perturbation has certain positive effects. The "GCN + LLM-based A-D" can achieve performance gains compared to the GCN (or GCN$_{\text{DeBERTa}}$) in most cases. This is attributed to the powerful text understanding ability of the LLM, which enables it to filter out unreliable edges in the graph and add reliable ones, thereby enhancing the quality of the learned representations (through


(a) Cora dataset.

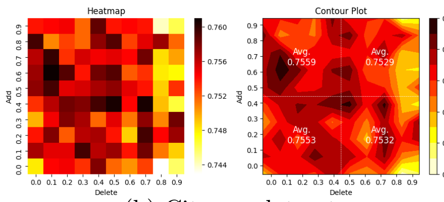
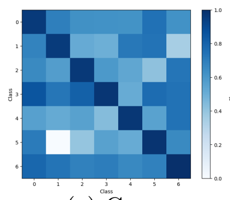
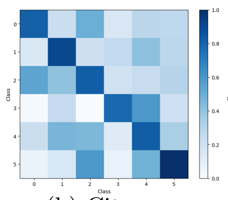
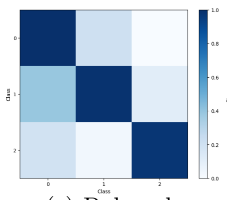

(b) Citeseer dataset.

Fig. 4: Heatmaps and contour plots on different datasets.


(a) Cora.


(b) Citeseer.


(c) Pubmed.

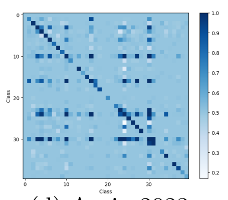

(d) Arxiv-2023.

Fig. 5: The relationship between the refined edge weights and classes.

facilitating the process of message passing). Similar performance gains can be observed in the "GCN + LLM-based LPA". Compared to the classical label propagation method like GCN-LPA$_{\text{DeBERTa}}$, the LLM-based pseudo-label propagation introduces more label information by utilizing the powerful reasoning capabilities of the LLM. Specifically, the GCN-LPA$_{\text{DeBERTa}}$ only uses the labels of labeled nodes (unlabeled nodes use a default label) during label propagation, whereas the "GCN + LLM-based LPA" not only uses the labels of labeled nodes, but also high-quality pseudo-labels generated by the LLM for unlabeled nodes. The additional label information aids label propagation, which in turn helps guide the model to learn more proper edge weights (i.e., a better graph topology). Although the LLM-based edge deletion/addition and the LLM-based pseudo-label propagation have differences in the direction of optimizing the graph topology, we find that combining the two can further enhance the model performance. As shown in the Table 1, the "GCN + LLM-based A-D & LPA" is almost always the top performer across all settings under the traditional paradigm. We also verify the generalizability of the refined graph topology structure on the SOTA paradigm TAPE, i.e., using the "TA+P+E" feature [8]. "TA+P+E" feature is a recently proposed LLM-augmented feature that incorporates the original features of the TAG (TA), LLM explanations (E), and LLM predictions (P). The results demonstrate that the LLM-based graph topology refinement approaches also benefit learning with the "TA+P+E" feature, showcasing the versatility of the proposed methods to some extent.

Few-shot setting. To further demonstrate the impact of pseudo-labels generated by the LLM on the label propagation, we test the performance of the "GCN + LLM-based LPA" in the few-shot setting, as shown in the Table 2. The "GCN + LLM-based LPA" has achieved performance gains in different scenarios, which is consistent with the experimental results under the general setting. We

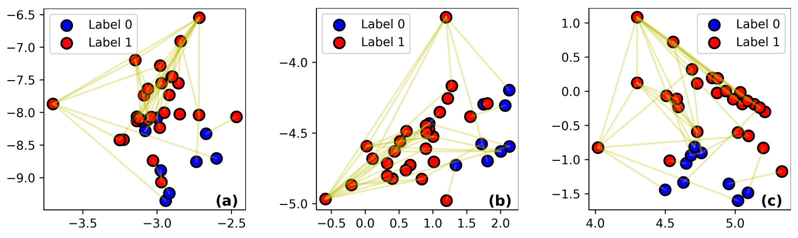
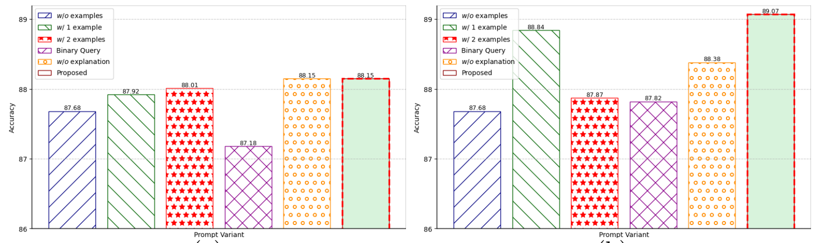


Fig. 6: Discriminativeness of learned embeddings. (a) w/o topology enhancement. (b) $w/$ LLM-based A-D. (c) $w/$ LLM-based LPA.



(a) (b)

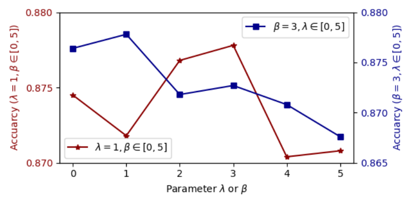
Fig. 7: Prompt variant analysis. (a) Traditional paradigm. (b) TAPE paradigm.

also test the generalizability of the optimized graph topology structure on the SOTA paradigm TAPE. The experimental results demonstrate that the LLM-based graph topology perturbation also improves learning with the "TA+P+E" feature under the few-shot setting.

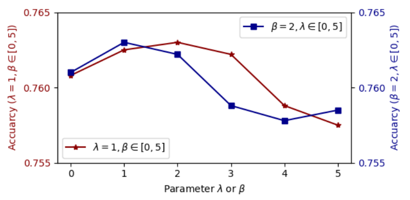
5.4 Visualization and Analysis

Visualization of LLM's topology refinement capability. In Fig. 4, we present heatmaps and contour plots on different datasets, which are obtained by adjusting the thresholds used in the edge deletion and addition processes. From these heatmaps and contour plots, we can observe a consistent pattern: the values in the top left corner of the figures are relatively large, while the values in the bottom right corner are relatively small. In other words, adding edges that LLM considers reliable and deleting those are considered as unreliable is beneficial to improving the topological structure, thereby improving the classification accuracy. For example, in the contour plot of the Cora, the average accuracy in the top left corner is 0.8754, while in the bottom right corner it is 0.8708. This phenomenon indicates that LLMs do have the ability to make reasonable edge additions/deletions based on the semantics of the input text.

Analysis of learned edge weights and node embeddings. In Fig. 5, we visualize the relationship between the refined edge weights and classes. Specifically, we divide the nodes into different groups according to their classes and then calculate the average edge weights between different groups. It is evident



(a) General setting.



(b) Few-shot setting.

Fig. 8: Parameter sensitivity of λ and β.

from Fig. 5 that the average weights on the diagonal are the highest. This indicates that the proposed model tends to strengthen the weights between nodes of the same class (i.e., emphasizing reliable edges in message passing) to achieve the purpose of topology enhancement. We also show the learned embeddings in Fig. 6. We sample nodes of two different classes from the Cora, and project the embeddings of these nodes onto a 2D plane using t-SNE. It can be observed that the embeddings with topology enhancement exhibit better discrimination, which aligns with the theoretical analysis (shrinking theory) provided in Section 4.5.

Sensitivity evaluation of prompts and parameters. In Fig. 7, we analyze different variants of prompts, including using fewer examples (w/o examples, $w/$ 1 example, and $w/$ 2 examples), directly asking the LLM to determine whether an edge should exist (YES or NO, binary query), and not requiring the LLM to provide an explanation for its decisions (w/o explanation). The experimental results indicate that using fewer examples degrades performance, for more examples help the LLM acquire task-specific knowledge. Additionally, binary queries and explanation-free modes are also detrimental to performance. This is because the fine-grained threshold queries and decision attributions in the proposed prompt template (Section 4) increase the reliability of the LLM's responses. Without them, the likelihood of LLM hallucinations will increase. The parameter sensitivity analysis is shown in Fig. 8. The phenomenon shown in Fig. 8 indicates that the modules corresponding to both hyperparameters have positive effects on the final model performance. The absence of either one ($\lambda = 0$ or $\beta = 0$) results in a decrease in model performance.

6 Conclusion

In this paper, we investigate the potential of LLMs in refining graph topology. First, we explore ways to leverage LLMs' capabilities in deleting unreliable edges and adding reliable edges in TAGs through the prompt statement. Second, considering LLMs' text understanding ability can help generate high-quality pseudo-labels, we propose to leverage the pseudo-labels produced by LLMs to improve the topological structure of TAGs. Finally, we integrate the above methods into the GNN training. The experimental results demonstrate that LLMs can play a positive role in perturbing the topological structure of graphs.

Acknowledgement. This work was supported by the NSFC Young Scientists Fund (No. 9240127), the Donations for Research Projects (No. 9229129) of the City University of Hong Kong, the Early Career Scheme (No. CityU 21219323), and the General Research Fund (No. CityU 11220324) of the University Grants Committee (UGC).

References

1. Brown, T., et al.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems (2020)
2. Chen, Z., Mao, H., Li, H., Jin, W., Wen, H., Wei, X., Wang, S., Yin, D., Fan, W., Liu, H., Tang, J.: Exploring the potential of large language models (llms) in learning on graphs. In: ACM SIGKDD Explorations Newsletter. vol. 25 (2024)
3. Chien, E., Chang, W.C., Hsieh, C.J., Yu, H.F., Zhang, J., Milenkovic, O., Dhillon, I.S.: Node feature extraction by self-supervised multi-scale neighborhood prediction. In: Proc. Int. Conf. Learning Representations (2022)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (2023)
5. Giles, C.L., Bollacker, K.D., Lawrence, S.: Citeseer: An automatic citation indexing system. In: Proc. ACM Conf. Digital Libraries (1998)
6. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems (2017)
7. He, P., Liu, X., Gao, J., Chen, W.: Deberta: Decoding-enhanced bert with disentangled attention. In: arXiv:2006.03654 (2020)
8. He, X., Bresson, X., Laurent, T., Perold, A., LeCun, Y., Hooi, B.: Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In: Proc. Int. Conf. Learning Representations (2024)
9. Hong, C., Liu, Z., Yang, J.: Sparsity induced similarity measure for label propagation. In: Proc. IEEE Int. Conf. Computer Vision (2009)
10. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: Datasets for machine learning on graphs. arXiv:2005.00687 (2020)
11. Hu, Y., Chen, C., Yang, C.H., Li, R., Zhang, D., Chen, Z., Chng, E.: Gentranslate: Large language models are generative multilingual speech and machine translators. In: ACL (2024)
12. Kaplan, J., McCandlish, S., Henighan, T., Brown, T., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling laws for neural language models. In: arXiv:2001.08361 (2020)
13. Karasuyama, M., Karasuyama, H.: Manifold-based similarity adaptation for label propagation. In: Advances in Neural Information Processing Systems (2013)
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proc. Int. Conf. Learning Representations (2016)
15. Lin, X., Kang, J., Cong, W., Tong, H.: Bemap: Balanced message passing for fair graph neural network. In: LoG (2023)
16. Liu, Y., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S.J., Yang, Y.: Learning to propagate labels: Transductive propagation network for few-shot learning. In: Proc. Int. Conf. Learning Representations (2019)
17. Mai, Z., Zhang, J., Xu, Z., Xiao, Z.: Financial sentiment analysis meets llama 3: A comprehensive analysis. In: MLMI (2024)

18. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Information Retrieval **3**, 127–163 (2000)
19. Miaschi, A., Dell'Orletta, F.: Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In: ACL (2020)
20. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: arXiv:1301.3781 (2013)
21. OpenAI: Gpt-4 technical report. In: arXiv:2303.08774 (2023)
22. Qin, Y., Wang, X., Zhang, Z., Zhu, W.: Disentangled representation learning with large language models for text-attributed graphs. In: arXiv:2310.18152 (2023)
23. Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., Huang, X.: Pre-trained models for natural language processing: A survey. Science China Technological Sciences **63**(10), 1872–1897 (2020)
24. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine **29**(3), 93–106 (2008)
25. Sun, S., Ma, C.: Hyperbolic contrastive learning with model-augmentation for knowledge-aware recommendation. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases (2024)
26. Sun, S., Yu, W., Ren, Y., Du, W., Liu, L., Zhang, X., Hu, Y., Ma, C.: Gdiffretro: Retrosynthesis prediction with dual graph enhanced molecular representation and diffusion generation. In: arXiv:2501.08001 (2025)
27. Tang, J., Yang, Y., Wei, W., Shi, L., Su, L., Cheng, S., Yin, D., Huang, C.: Graphgpt: Graph instruction tuning for large language models. In: arXiv:2310.13023 (2023)
28. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models. In: arXiv:2302.13971 (2023)
29. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: Proc. Int. Conf. Learning Representations (2018)
30. Wang, H., Leskovec, J.: Combining graph convolutional neural networks and label propagation. ACM Trans. Information Systems **40**(4), 1–27 (2021)
31. Wang, Z., Wang, R., Wang, M., Lai, T., Zhang, M.: Self-supervised transformer-based pre-training method with general plant infection dataset. In: PRCV (2024)
32. Xiao, Z., Blanco, E., Huang, Y.: Analyzing large language models' capability in location prediction. In: LREC-COLING (2024)
33. Yang, M., Meng, Z., King, I.: L2 feature normalization for dynamic graph embedding. In: Proc. IEEE Int. Conf. Data Mining (2020)
34. Zhao, H., Liu, S., Ma, C., Xu, H., Fu, J., Deng, Z.H., Kong, L., Liu, Q.: Gimlet: A unified graph-text model for instruction-based molecule zero-shot learning. In: arXiv:2306.13089 (2023)
35. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems (2004)
36. Zhou, D., Schölkopf, B.: Regularization on discrete spaces. In: DAGM Symposium (2005)
37. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: Proc. Int. Conf. Machine Learning (2003)
38. Zhu, Y., Wang, Y., Shi, H., Tang, S.: Efficient tuning and inference for large language models on textual graphs. In: arXiv:2401.15569 (2024)