Let's Ask GNN: Empowering Large Language Model for Graph In-Context Learning

Zhengyu Hu[1*], **Yichuan Li**[2*], **Zhengyu Chen**[3], **Jingang Wang**[3], **Han Liu**[1], **Kyumin Lee**[2], **Kaize Ding**[1],
[1]Northwestern University, [2]Worcester Polytechnic Institute, [3]MeiTuan
Correspondence: kaize.ding@northwestern.edu

Abstract

Textual Attributed Graphs (TAGs) are crucial for modeling complex real-world systems, yet leveraging large language models (LLMs) for TAGs presents unique challenges due to the gap between sequential text processing and graph-structured data. We introduce AskGNN, a novel approach that bridges this gap by leveraging In-Context Learning (ICL) to integrate graph data and task-specific information into LLMs. AskGNN employs a Graph Neural Network (GNN)-powered structure-enhanced retriever to select labeled nodes across graphs, incorporating complex graph structures and their supervision signals. Our learning-to-retrieve algorithm optimizes the retriever to select example nodes that maximize LLM performance on graph. Experiments across three tasks and seven LLMs demonstrate AskGNN's superior effectiveness in graph task performance, opening new avenues for applying LLMs to graph-structured data without extensive fine-tuning.

1 Introduction

Textual attributed graphs (TAGs) (Chen et al., 2024b,c; Hu et al., 2020) are pivotal in modeling complex real-world systems, from social networks to recommendation engines and information retrieval systems. In TAGs, nodes represent text documents, while edges depict relationships between them like shown in Figure 1. The interconnected nature of TAGs encapsulates rich knowledge and context, offering a more comprehensive understanding of the underlying data structures than isolated text analysis alone.

Recent advancements in large language models (LLMs) (Brown et al., 2020; Dong et al., 2024; Hu et al., 2024) have demonstrated remarkable zero-shot and few-shot capabilities across a wide range of tasks. These LLMs excel in areas such as data augmentation (Li et al., 2024b), text summarization (Jin et al., 2024b), and content recommendation (Wu et al., 2024). However, despite their impressive capabilities, LLMs usually face significant limitations in processing and leveraging the structural information inherent in TAGs (Yasunaga et al., 2022; Fatemi et al., 2023). Trained mainly on unstructured text data, these LLMs lack the innate ability to interpret and utilize the structural information that are fundamental to TAGs. This limitation hinders their effectiveness in tasks that require a deeper understanding of interconnections among text documents.

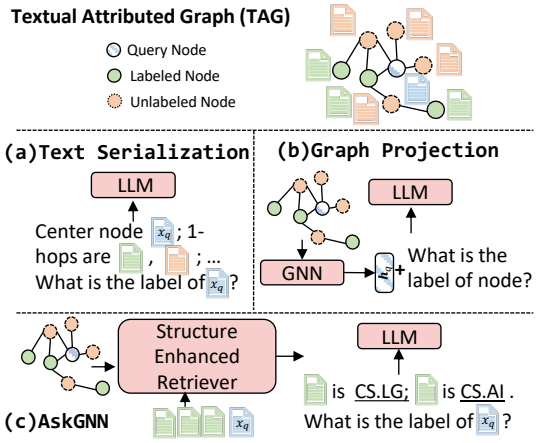

Figure 1: Illustration of methods utilizing LLMs for graph tasks, with a focus on node classification. Our proposed method enhances structure and task understanding by retrieving insightful examples that improve the LLMs' comprehension of graph complexities.

To bridge this gap and empower LLMs for data-efficient graph learning (Zhang et al., 2022; Ding, 2024), researchers have developed two primary approaches to integrate structural information into LLM understanding. One line of research, exemplified by studies such as Fatemi et al. (2023) and Ye et al. (2024), employs text-based serialization methods to encode k-hop neighbors as the

[*]These authors contributed equally. This work was mainly done during Zhengyu's internship at Northwestern.

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 1396–1409
November 12-16, 2024

context around the query node for LLMs (as illustrated in Figure 1 (a)). However, this method often overlooks nodes beyond the k-hop neighborhood, missing critical signals and introducing noise, leading to underperformance compared to Graph Neural Networks (GNNs) in tasks like node classification (Chen et al., 2024b; Fatemi et al., 2023). Another line of research, explored by Chen et al. (2024a); Perozzi et al. (2024); He et al. (2024), employs GNNs to encode TAGs into compact graph tokens (Fig. 1 (b)). While this approach preserves structural information through the message-passing mechanism, it raises the modality misalignment issue between graph structure and text spaces, impeding effective integration with LLMs. These limitations underscore the need for a new solution that can better elicit the power of LLMs on TAGs.

In this paper, we propose AskGNN, a novel framework that transforms graph structural information and task supervision signals into a limited number of document node-label pairs. This approach benefits from LLMs' exceptional power in In-Context Learning (ICL) (Kaplan et al., 2020; Brown et al., 2020; Li et al., 2024c; Dong et al., 2024), leveraging both graph structural information and textual information without extensive fine-tuning. At the core of AskGNN is a GNN-based structure-enhanced retriever, designed to select the most relevant document node-label pairs as ICL examples for LLM few-shot prompting. This strategy harnesses GNNs' prowess in extracting structural information while circumventing the semantic gaps between text and non-text tokens (Ding et al., 2020, 2022; Hu et al., 2023). To optimize the retriever, we further introduce a learning-to-retrieve algorithm that fine-tunes the structure-enhanced GNN retriever, ensuring the selection of the most pertinent ICL node examples. By integrating these components, AskGNN offers a new solution that addresses the limitations of previous approaches, enabling LLMs to effectively process and utilize the rich structural and supervisory information inherent in TAGs, while leveraging the adaptability of ICL to diverse tasks with limited supervision, as depicted in Figure 1(c). To summarize, our main contributions are as follows:

- We introduce AskGNN, a new graph in-context learning framework that empowers LLMs for few-shot learning on TAGs.
- We develop a learning-to-retrieve algorithm that enhances the retriever's ability to retrieve examples that are both structurally informative and contextually relevant for LLMs' ICL tasks.
- We conduct extensive evaluations across three distinct tasks and seven LLMs, demonstrating the superior robustness and effectiveness of AskGNN in improving graph task performance.

2 Related Work

LLMs for Graph Learning. Applying LLMs to text-attributed graph tasks provides significant opportunities for advancing data-efficient graph learning (He et al., 2023; Li et al., 2023b; Ding, 2024; Jin et al., 2024a; He et al., 2024). These tasks require LLM to process and understand the complex graph structural information, a challenge that has only recently gained attention (Wang et al., 2024; Ye et al., 2024; Huang et al., 2024; He et al., 2024). Some approaches enable LLMs to learn structural information by tuning their parameters (Ye et al., 2024) or through graph instruction tuning (Tang et al., 2024). Others transform graphs into hidden embeddings to aid in understanding relational data (He et al., 2024; Ye et al., 2024; Chen et al., 2023). However, these methods either cannot fully capture informative graph structural information or align the graph and text tokens.

In-Context Learning. ICL (Dong et al., 2024; Xie et al., 2022) allows pretrained LLMs to make predictions for diverse downstream tasks by directly prompting them with a few examples of the task or textual instructions (Pathak et al., 2016; Min et al., 2016)without finetuning. Jiang et al. (2023) improves language models by iteratively retrieving relevant information during text generation, boosting performance in long-form tasks. Asai et al. (2024)enhances language models through adaptive retrieval and self-reflection, outperforming state-of-the-art models in various tasks. Rubin et al. (2022); Li et al. (2023a); Wu et al. (2023) train a separate retriever that selects relevant examples using a unified ranking framework, outperforming task-specific methods. This capability significantly reduces the adaptation effort compared to traditional fine-tuning approaches and has shown robust performance across a variety of models and tasks (Dong et al., 2024; Rubin et al., 2022; Li et al., 2024c). Unlike models such as BERT (Devlin et al., 2019), which require extensive fine-tuning for new tasks, LLMs leverage their pretrained knowledge effectively through in-context prompts (He et al., 2024; Ye et al., 2024).

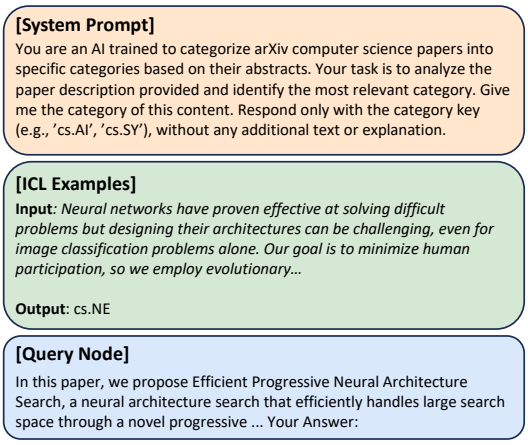

Figure 2: Example of node classification task, illustrating how retrieved ICL examples are integrated with the query node with the prompt.

3 Problem Definition

In this work, we define a TAG as $\mathcal{G} = (\mathbf{A}, \mathcal{X})$, where $\mathbf{A} \in \{0,1\}^{N\times N}$ represents the adjacency matrix, with $\mathbf{A}_{ij} = 1$ indicating a connection between nodes i and j, N is the total number of nodes and $\mathcal{X} = \{x_i\}_{i=0}^{N}$ is text corpus for each node. Each node i is associated with a text document represented as a sequence of tokens $x_i = \{w_v\}_{v=0}^{|x_i|}$. Among the nodes in TAG, only a subset are labeled, denoted as $\mathcal{D} = \{(x_i, y_i)\}_{i=0}^{M}$, where $M \ll N$ and $y_i = \{w_v\}_{v=0}^{|y_i|}$ represents the text labels of node i (e.g., 'cs.LG' in the context of arxiv classification).

Our objective is to encode both the graph structural information and task supervision signals by retrieving a subset of K labeled examples, $\mathcal{D}_q = \{(x_i, y_i)\}_{i\in[K]}$ from $\mathcal{D}$. These examples serve as context for prompting LLMs to address graph-specific tasks for a query node q. This retrieval process, constrained by both context length and computational costs, selects far fewer examples than the number of labeled nodes, with $K \ll N$. The retrieval and prompting process is formalized as follows:

$$\mathcal{D}_q = R(x_q, \mathcal{D}, \mathcal{G}), \tag{1}$$

$$\hat{y}_q = \text{LLM}(T(\mathcal{D}_q, x_q)), \tag{2}$$

where T is the prompt template used to encode both the text from the labeled nodes in $\mathcal{D}_q$ and the query node x_q. An illustrative example for the node classification task is provided in Figure 2.

4 Proposed Approach – AskGNN

We present AskGNN, a structure-enhanced framework designed to optimize LLMs for graph ICL. The whole framework is illustrated in Figure 3. This section details the key components: the GNN-based structure-enhanced retriever (Section 4.1), LLM feedback collection (Section 4.2), retriever optimization (Section 4.3), and the utilization of the optimized retriever for ICL example selection (Sectio 4.4).

4.1 Structure-Enhanced Retriever

The Structure-Enhanced Retriever (SE-Retriever) plays a key role in utilizing GNNs to enhance the ICL process for LLMs. Previous methods based solely on text similarity (Lewis et al., 2020; Karpukhin et al., 2020; Xiong et al., 2021) fall short when applied to TAGs, overlooking valuable structural information. The SE-Retriever is designed to select the most relevant ICL examples from a graph-structured dataset, leveraging both semantic and structural information. The retrieval process starts with the GNN extracting feature representations from the nodes. For a query node x_q, its representation at the l-th GNN layer is $\mathbf{h}_q^l = \text{GNN}(\mathbf{h}_q^0, \mathbf{A}, \mathbf{H}^0)$, with $\mathbf{h}_q$ denoting the final layer output. The retriever identifies the top K labeled nodes from M labeled nodes based on cosine similarity:

$$\{(x_k, y_k)\}_{k=1}^{K} = \text{TopK}_{i\in[M]}\, \text{sim}(\mathbf{h}_q, \mathbf{h}_i). \tag{3}$$

The selected ICL examples for the query are formalized as:

$$\mathcal{D}_q = \{(x_i, y_i)\}_{i\in[K]}. \tag{4}$$

The SE-Retriever ensures that, for each query x, the top K samples with the highest cosine similarity are selected, effectively incorporating both textual and structural information into the ICL example selection process.

4.2 Learning-to-Retrieve via LLM Feedback

We propose a novel Learning-to-Retrieve (L2R) approach that leverages LLM feedback as a training signal to optimize the retriever. This method establishes a dynamic learning loop that continuously refines the retriever's selection of examples based on LLM feedback. The following subsections detail the components and implementation of this learning process.

ICL Training Example Curation. Our approach, AskGNN aims to identify which K labeled nodes from the TAGs can serve as effective context

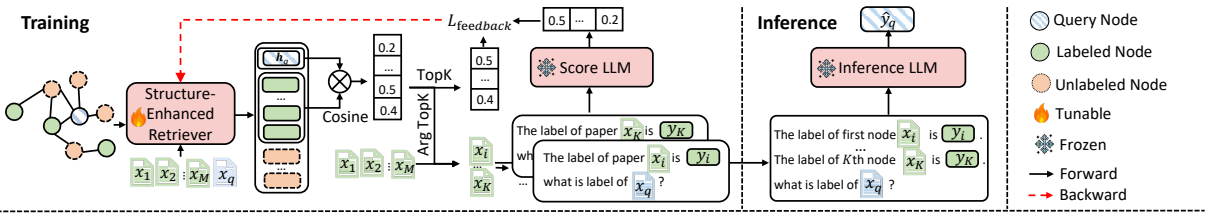

Figure 3: The overall framework of AskGNN, illustrating the structure-enhanced retriever based on GNNs for selecting ICL examples. The framework integrates LLM feedback to optimize the retriever, improving its ability to select relevant examples for graph-based tasks.

for the query node in the LLM. However, evaluating all $K!$ permutations of examples for each query node x_q is computationally prohibitive. To address this, we decompose the problem into evaluating individual examples and collect LLM feedback for each ICL example separately. To further reduce computational costs and iteratively improve GNN retriever, we only collect feedback for the top-K most similar nodes to each query node, caching the results for efficiency.

LLM Feedback Quantification. We introduce a novel "utility score" based on the inverse of perplexity (PPL) to quantify the contribution of a selected example towards the LLM's correct prediction. The utility score for an ICL example $e = (x_e, y_e)$ is defined as:

$$s(e) = \frac{\frac{1}{\text{PPL}(y_q)}}{\sum_{y_c \in \mathcal{Y}} \frac{1}{\text{PPL}(y_c)}}, \tag{5}$$

where $\mathcal{Y}$ represents the set of all possible classes. The perplexity for a query x_q with respect to a candidate class y_c and an ICL example e is computed as:

$$\text{PPL}(y_c) = \exp\left\{-\frac{1}{|y_c|}\sum_{v\in[|y_c|]} \log p\left(w_v \mid w_{<v}, x_q, e\right)\right\}. \tag{6}$$

This approach allows us to evaluate the contribution of each sample in $\mathcal{D}_q$, capturing the LLM's implicit feedback on the effectiveness of different ICL examples.

ICL Example Ranking and Feedback Loss. We rank the ICL examples in descending order based on their utility scores:

$$\hat{\mathcal{D}}_q = \text{Desc.}(s(e) \mid e \in \mathcal{D}_q) \tag{7}$$

The resulting ranked set $\hat{\mathcal{D}}_q$ serves as a training signal to optimize the GNN retriever, effectively incorporating LLM feedback into the retrieval process. To achieve this optimization, we define a loss function $\mathcal{L}_{\text{feedback}}$ that focuses on retrieving optimal ICL examples:

$$\mathcal{L}_{\text{feedback}} = -\sum_{q\in[M]}\sum_{k\in[K]} \log \frac{e^{\text{sim}(\mathbf{h}_q,\mathbf{h}_k)}}{\sum_{j=0}^{K} e^{\text{sim}(\mathbf{h}_q,\mathbf{h}_j)}}, \tag{8}$$

where k represents the k-th example in $\hat{\mathcal{D}}_q$.

4.3 Optimization

The final loss function combines two components:

$$\mathcal{L} = \beta \times \mathcal{L}_{\text{feedback}} + (1-\beta) \times \mathcal{L}_{\text{clf}}, \tag{9}$$

where $\mathcal{L}_{\text{feedback}}$ focuses on retrieving optimal ICL examples, and $\mathcal{L}_{\text{clf}}$ targets the graph learning. The $\mathcal{L}_{\text{clf}}$ is the node classification loss, defined as:

$$\mathcal{L}_{\text{clf}} = -\sum_{i\in[N]} \text{Cross-Entropy}\left(x_i, y_i\right). \tag{10}$$

Optimizing $\mathcal{L}_{\text{clf}}$ helps the SE-Retriever learn both local and global structural patterns, complementing the $\mathcal{L}_{\text{feedback}}$ to improve example selection and task performance.

4.4 Model Inference

ICL Example Selection. The SE-Retriever R is employed to select an optimal set of ICL examples from the labeled TAGs. This selection is driven by the learned structural information and the similarity between the query x_q and the potential examples:

$$\{(x_i, y_i)\}_{i\in[K]} = R(x_q, \mathcal{D}, \mathcal{G}). \tag{11}$$

Node Classification. The LLM leverages the selected ICL examples $\hat{\mathcal{D}}_q = \{(x_i, y_i)\}_{i\in[K]}$ alongside the query x_q to perform the classification task. This process is formalized as follows:

$$\hat{y}_q = \text{LLM}(T(\hat{\mathcal{D}}_q, x_q)), \tag{12}$$

where T is the prompt template used to encode both the labeled nodes in $\hat{\mathcal{D}}_q$ and the text of the query

node x_q. By integrating the most relevant examples, this approach enhances the LLM's understanding, resulting in improved prediction accuracy for our tasks.

5 Experiment

5.1 Experimental Setup

Evaluation Datasets. In this paper, we adopt the following TAG datasets widely used for node classification: ogbn-arxiv, ogbn-products (OGB) (Hu et al., 2020) and arxiv2023 (He et al., 2023), The statistics and descriptions of these datasets are provided in Appendix C. To assess the effectiveness of AskGNN under low-data scenarios, we limit the training dataset to 1%, 5%, and 10% of the labeled nodes. We use Accuracy as the evaluation metric in all experiments and employ the default numerical node embeddings as document node representation.

Baseline Methods. The baseline methods in our study fall into the following categories: *(1)* Bare GNNs: GCN (Kipf and Welling, 2017), GraphSAGE (Hamilton et al., 2017) and GraphSAINT (Zeng et al., 2020). *(2)* Text-based serialization: *K-Hop* (Chen et al., 2024b), *Graph-CoT* (Jin et al., 2024a)); *(3)* Graph projection: *InstructGLM* (Ye et al., 2024); *(4)* ICL: Zero-shot, randomly sampling based few-shot and semantic-based k-NN few-shot; *(5)* InstructTuning (Zheng et al., 2024). To show the superiority of our approach, we first include both widely used and state-of-the-art methods as our baselines. A detailed description of these methods can be found in Appendix A.

Implementation Details. We adopted Qwen1.5-72B (Bai et al., 2023) and Llama3-70B (AI@Meta, 2024) as our primary inference LLMs in the main experiments. For scoring, we utilize smaller models like Qwen1.5-7B and Llama-8B to complement the larger inference LLMs. Additional model architectures, such as Mistral-8x7B (Mistralai, 2024), and varying model scales, including 7B, 14B, and 32B, are tested in Section 6.1. We use GraphSAGE (Hamilton et al., 2017) as our retriever backbone.

5.2 Experiment Results

Main Results. To evaluate the overall performance of AskGNN, we conducted experiments across three datasets. The results presented in Table 1 highlight several key findings: First, our approach consistently outperforms other methods across all datasets and baselines, demonstrating its effectiveness in selecting optimal examples for ICL. Second, the correct selection of ICL examples significantly enhances LLM performance, whereas poorly chosen ICL examples can have detrimental effects. For instance, with the Llama3-70B model, Few-Shot learning performs worse than Zero-Shot. However, when the ICL examples are selected using AskGNN, the model's performance improves markedly. Similar conclusions are drawn for Few-shot (Rand.) and Few-shot (k-NN). Third, we observe that the impact of Instruct Tuning is correlated with the power of the LLM. Specifically, when the LLM is more powerful (e.g., Llama3-70B), the gains from Instruct Tuning are smaller compared to those observed with weaker models (e.g., Qwen1.5-72B). This may be because Llama3 has undergone extensive training on relevant tasks (AI@Meta, 2024). Finally, consistent with previous studies (Chen et al., 2024b), text-based serialization methods do not show significant improvement in text-attributed graph tasks compared to Zero-Shot methods. In graph datasets, these methods are less effective than ICL methods that effectively leverage structural information.

Extended Results. We expanded the evaluation of AskGNN to diverse tasks such as link prediction and conditional text generation, demonstrating its broader applicability and versatility in addressing various graph-based challenges. The results of link prediction and conditional text generation can be found in Figure 4. For the link prediction task, we reformulated the problem as binary question classification. In this setting, given a pair of nodes, the model predicts whether a connection exists between them. In our experiments, AskGNN outperformed baseline methods, achieving 89.06% accu-

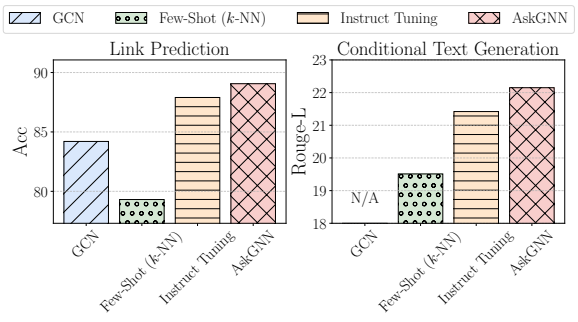

Figure 4: Performance of AskGNN on different tasks, including link prediction and conditional text generation.

Table 1: Averaged node classification accuracy on the `ogbn-arxiv`, `ogbn-products`, and `arxiv2023` datasets with 1%, 3%, and 10% of labeled nodes. For each LLM family, the best results are **bold**.

	Methods	ogbn-arxiv			ogbn-products			Arxiv2023			Avg.
		1%	5%	10%	1%	5%	10%	1%	5%	10%	
N/A	GCN	57.56 ±4.18	65.91 ±1.65	67.77 ±1.75	68.21 ±1.75	73.06 ±1.50	74.45 ±1.45	53.41 ±3.42	61.91 ±1.60	64.66 ±1.48	65.22
	GraphSAGE	57.72 ±1.66	62.20 ±2.23	65.71 ±2.33	65.72 ±1.66	71.32 ±1.80	72.45 ±1.75	54.67 ±2.14	61.71 ±1.50	64.51 ±1.50	64.00
	GraphSAINT	58.03 ±2.17	62.73 ±3.03	65.92 ±2.78	66.13 ±1.53	71.73 ±1.23	73.54 ±1.18	54.92 ±2.07	61.95 ±1.79	64.84 ±1.94	64.42
Qwen1.5-72B	K-Hop	62.21 ±0.22	62.21 ±0.22	62.21 ±0.22	66.85 ±0.15	66.85 ±0.15	66.85 ±0.15	67.47 ±0.24	67.47 ±0.24	67.47 ±0.24	65.29
	Graph-CoT	59.73 ±0.21	59.73 ±0.21	59.73 ±0.21	67.78 ±0.17	67.78 ±0.17	67.78 ±0.17	64.89 ±0.13	64.89 ±0.13	64.89 ±0.13	64.13
	InstructTuning	65.33 ±0.35	70.63 ±0.76	71.47 ±0.58	73.70 ±0.53	80.16 ±0.41	81.99 ±0.26	69.32 ±0.23	70.72 ±0.33	71.23 ±0.25	72.72
	InstructGLM	**65.55** ±0.26	**70.81** ±0.62	**71.51** ±0.36	73.83 ±0.48	**80.28** ±0.51	**82.12** ±0.24	69.31 ±0.19	**70.86** ±0.28	**71.31** ±0.21	**72.84**
	Zero-Shot	61.63 ±0.30	61.63 ±0.30	61.63 ±0.30	62.85 ±0.27	62.85 ±0.27	62.85 ±0.27	67.76 ±0.53	67.76 ±0.53	67.76 ±0.53	64.08
	Few-Shot (Rand.)	61.90 ±0.02	62.30 ±0.14	62.17 ±0.31	66.80 ±0.04	67.18 ±0.27	67.62 ±0.08	67.56 ±0.23	67.66 ±0.69	68.16 ±0.23	65.70
	Few-Shot (k-NN)	63.47 ±0.37	64.77 ±0.12	64.87 ±0.35	74.58 ±0.39	75.21 ±0.21	76.48 ±0.22	67.47 ±0.57	68.06 ±0.09	68.86 ±1.52	69.30
	AskGNN	64.67 ±0.10	66.95 ±0.23	67.82 ±0.16	**76.79** ±0.26	78.72 ±0.19	79.91 ±0.23	**69.56** ±0.22	69.97 ±0.12	70.46 ±0.17	**71.65**
Llama3-70B	K-Hop	66.27 ±0.28	66.27 ±0.28	66.27 ±0.28	75.93 ±0.29	75.93 ±0.29	75.93 ±0.29	68.23 ±0.19	68.23 ±0.19	68.23 ±0.19	70.14
	Graph-CoT	64.51 ±0.21	64.51 ±0.21	64.51 ±0.21	77.52 ±0.13	77.52 ±0.13	77.52 ±0.13	64.31 ±0.36	64.31 ±0.36	64.31 ±0.36	68.84
	InstructTuning	68.30 ±0.10	71.10 ±0.30	71.07 ±0.67	72.20 ±0.89	81.87 ±0.27	82.42 ±0.21	68.96 ±0.24	69.43 ±0.24	69.52 ±0.22	72.76
	InstructGLM	68.35 ±0.16	**71.17** ±0.25	71.13 ±0.42	72.19 ±0.77	81.70 ±0.29	82.59 ±0.32	69.02 ±0.17	69.57 ±0.13	69.61 ±0.32	72.82
	Zero-Shot	68.43 ±0.31	68.43 ±0.31	68.43 ±0.31	77.00 ±0.16	77.00 ±0.16	77.00 ±0.16	69.03 ±0.22	69.03 ±0.22	69.03 ±0.22	71.48
	Few-Shot (Rand.)	66.27 ±0.21	66.10 ±0.17	66.20 ±0.17	76.87 ±0.18	76.86 ±0.21	77.00 ±0.26	68.21 ±0.23	68.35 ±0.16	68.18 ±0.28	70.44
	Few-Shot (k-NN)	68.23 ±0.25	67.80 ±0.10	67.63 ±0.35	78.76 ±0.39	79.00 ±0.10	79.19 ±0.18	68.36 ±0.17	68.47 ±0.14	68.25 ±0.18	71.74
	AskGNN	**69.13** ±0.24	70.29 ±0.25	**71.53** ±0.11	**81.35** ±0.23	**82.54** ±0.13	**82.88** ±0.17	**73.86** ±0.22	**74.07** ±0.12	**74.97** ±0.17	**73.86**

racy, surpassing both GCN and instruction-tuned models. In the conditional text generation task, we evaluated AskGNN's ability to integrate graph knowledge into language generation. The model was given with 10% of a query node's text and tasked with generating the remaining 90%. Using Rouge-L (Lin, 2004) as the evaluation metric, AskGNN achieved a score of 22.15, surpassing both few-shot learning and instruction-tuned baselines. These results underscore AskGNN's flexibility and adaptability in many other tasks besides node classification.

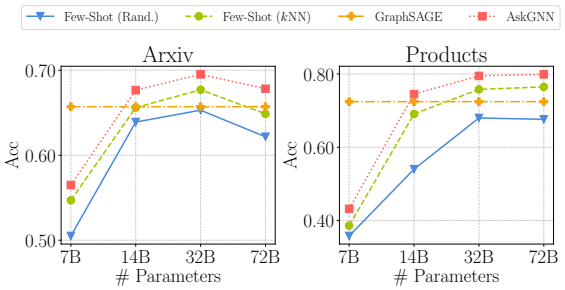

Figure 5: Performance of the Qwen1.5 family across parameter sizes (7B to 72B) on ogbn-product and ogbn-arxiv datasets.

6 Further Analysis

6.1 Analysis on LLMs' Sizes and Families

This section aims to provide a comprehensive understanding of how varying inference LLM's sizes and architectures impact the performance of AskGNN in integrating semantic and structural information from text-attributed graphs.

Same Family but Different Sizes. As shown in Figure 5, we explore the scalability of the Qwen1.5 (Bai et al., 2023) architecture on the ogbn-product and ogbn-arxiv datasets, with model scales ranging from 7B to 72B parameters. The empirical results indicate that AskGNN consistently outperforms both few-shot (k-NN) and few-shot (Rand.), highlighting the effectiveness of selecting structure ICL examples across all model scales. Interestingly, we observe an inverse scaling phenomenon, where performance initially rises but then declines as the model size increases from 7B to 72B parameters. This behavior, seen in both datasets, contradicts the conventional expectation that larger models should inherently yield better performance. For ogbn-product, the 72B model shows lower accuracy compared to the 32B model, and a similar trend is observed for ogbn-arxiv This aligns with previous findings (McKenzie et al., 2023; Dohmatob et al., 2024), which suggest that inverse scaling behavior arises from misleading few-shot demonstrations, where larger models capture nuanced language aspects but are also misled by noise or non-representative features.

Different LLM Families. We tested our framework across various LLM families, including Dense models like Qwen1.5-72B and Llama3-70B, as well as Mixture of Experts (MoE) models like

Table 2: Comparison of performance between Majority Voting (MV), Few-shot (k-NN), and AskGNN across various experimental settings. MV selects the most frequent class in the ICL examples and consistently underperforms compared to LLM predictions, demonstrating the LLM's ability to make more nuanced decisions beyond simple class frequency.

Methods	ogbn-arxiv			
	1%	3%	5%	10%
Few-Shot (k-NN)	61.90± 0.02	62.06± 0.11	62.30± 0.14	62.17± 0.31
MV+k-NN	29.97 ±0.32	35.50 ±0.82	34.10 ±0.10	35.33 ±0.12
AskGNN	69.13± 0.24	69.42± 0.12	70.29± 0.25	71.53± 0.11
MV+AskGNN	57.27± 0.21	59.37± 0.17	58.00± 0.10	60.87± 0.40
	ogbn-products			
	1%	3%	5%	10%
Few-Shot (k-NN)	66.80± 0.04	67.38± 0.19	67.18± 0.27	67.62± 0.08
MV + k-NN	47.30 ±0.44	54.69 ±0.31	54.09 ±0.72	53.01 ±0.31
AskGNN	65.24± 0.20	68.24± 0.19	82.54± 0.13	82.88± 0.17
MV + AskGNN	65.24± 0.20	68.24± 0.21	71.39± 0.39	71.63± 0.31

Mistral-8x7B. The results are shown in Figure 6. Our results show that AskGNN consistently outperforms Few-Shot, emphasizing the importance of selecting ICL examples across different model architectures, whether Dense or MoE. Notably, for less powerful models, optimal ICL example selection leads to greater performance gains, with a larger gap between AskGNN and Few-shot (k-NN) compared to Few-shot (Rand.). This highlights the significant impact of example selection on enhancing the capabilities of smaller or weaker models. For weaker models (e.g., Qwen1.5-72B), instruct tuning surpasses AskGNN. However, for stronger models like Llama3-70B, AskGNN proves more effective, likely due to Llama3-70B's training on more relevant tasks. Additionally, the performance gap between instruct tuning and AskGNN narrows under the MoE architecture. This is likely due to MoE's dynamic allocation capabilities, which enhance flexibility and efficiency in processing diverse inputs, leading to better integration of instruct tuning and ICL, thereby reducing the disparity between the two methods.

6.2 Hyperparameter Analysis on β

To investigate the impact of the classification loss $\mathcal{L}_{\text{clf}}$ introduced in Equation 9, we conducted a comprehensive hyperparameter analysis on the weighting factor β. Figure 7 illustrates the performance trends as β varies from 0.0 to 1.0 on both the ogbn-arxiv and ogbn-products datasets using the Qwen1.5-72B model. For ogbn-arxiv, we observe peak performance at $\beta \approx 0.2$, suggesting a balanced contribution from both feedback-based and structural-based losses maximizes performance. In contrast, ogbn-products exhibits higher sensitivity to β, peaking at $\beta = 0.3$ with a steeper subsequent decline, indicating a greater dependence on the structural component. We hypothesize this is due to the complex nature of product descriptions benefiting more from structural insights. These observations underscore the importance of fine-tuning the balance between different components of the loss function tailored to the specific characteristics of the dataset, thereby enabling optimal model performance. These observations underscore the importance of fine-tuning the balance between different components of the loss function tailored to the specific characteristics of the dataset, thereby enabling optimal model performance.

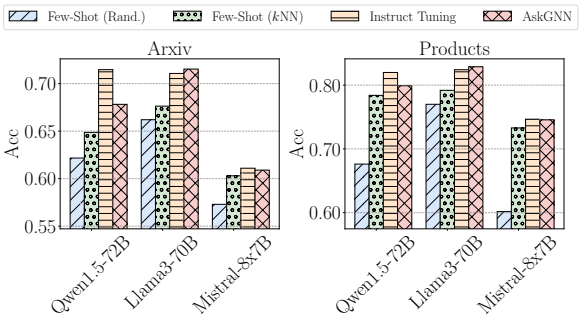

Figure 6: Performance comparison of AskGNN across different LLM architectures, including Dense models (Qwen1.5-72B, Llama3-70B) and MoE models (Mistral-8x7B).

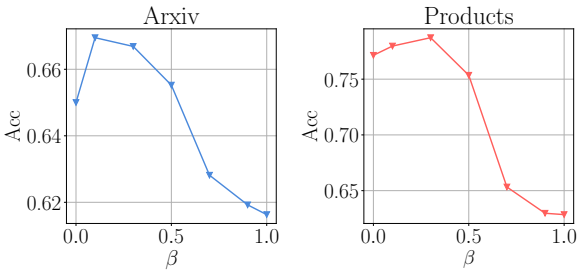

Figure 7: Performance variation with respect to hyperparameter β on ogbn-arxiv and ogbn-products datasets using Qwen1.5-72B.

6.3 Analysis on ICL Examples

Cast Study. We provide a case study of retrieved examples from the ogbn-product dataset to illustrate the difference in example selection between AskGNN and k-NN, as shown in Table 4. This case study demonstrates that while k-NN selects se-

Table 3: Experiment results of different heuristic ICL examples retrieve appraoches. The best results are **bold**.

	Methods	ogbn-arxiv			ogbn-products			Arxiv2023			Avg.
		1%	5%	10%	1%	5%	10%	1%	5%	10%	
wen1.5 72B	*NPL*	63.91 ±0.26	63.91 ±0.26	63.91 ±0.26	73.83 ±0.17	73.83 ±0.17	73.83 ±0.17	63.07 ±0.83	65.78 ±0.69	65.82 ±0.43	67.54
	NPG	62.43 ±0.38	66.47 ±0.29	65.90 ±0.33	75.23 ±0.34	76.62 ±0.26	77.89 ±0.17	62.17 ±0.23	63.46 ±0.17	64.62 ±0.13	68.42
	AskGNN	**64.67** ±0.10	**66.95** ±0.23	**67.82** ±0.16	**76.79** ±0.26	**78.72** ±0.19	**79.91** ±0.23	**69.56** ±0.22	**69.97** ±0.12	**70.46** ±0.17	71.65
lama3 70B	*NPL*	**69.49** ±0.35	69.49 ±0.35	69.49 ±0.35	80.86 ±0.18	80.86 ±0.18	80.86 ±0.18	72.27 ±0.19	72.27 ±0.19	72.27 ±0.19	73.65
	NPG	65.50 ±0.20	68.57 ±0.21	68.33 ±0.21	77.61 ±0.06	79.03 ±0.16	80.22 ±0.27	67.77 ±0.22	70.35 ±0.14	71.08 ±0.19	71.16
	AskGNN	69.13 ±0.24	**70.29** ±0.25	**71.53** ±0.11	**81.35** ±0.23	**82.54** ±0.13	**82.88** ±0.17	**73.86** ±0.22	**74.07** ±0.12	**74.97** ±0.17	**75.62**

mantically similar examples (focusing on underwater themes), it misses the crucial "Toy" classification. In contrast, AskGNN retrieves an example that, despite having lower semantic similarity, correctly captures the "Toy" category. This highlights AskGNN's ability to prioritize task-relevant information over semantic similarity, resulting in more accurate classifications for the LLM.

Table 4: Comparison of retrieved examples from the ogbn-product dataset. **Bold** words indicate semantic similarity.

Type	Text	Label
Query	wonder **pets flyboat** with 3 removable figures for everyday fun	Toy
k-NN	dive into adventure with your favorite **underwater** explorers, the **octonauts**! on this exciting dvd, captain barnacles tangles with a colossal squid	Movie
AskGNN	choose your favorite paw patrol character and rush into adventure bay action with their special vehicle!	Toy

Comparison with 1-hop Neighbors with Pseudo Labels. To understand the importance of retrieve ICL examples across whole graph, we consider utilize the query node's 1-hop neighbors. Huang et al. (2023) suggests that leveraging a query's neighbor and its label can yield satisfactory results. However, since actual labels for the query nodes' neighbors are not accessible, we design two approaches for getting the pseudo label: NPL and NPG, NPL stands for Neighbors Pseudo Labels generated by the LLM, while NPG stands for Neighbors' Pseudo Labels generated by the GNN. Results in Table 3 show that AskGNN surpasses *GE*, *NPL*, and *NPG*, highlighting the importance of retrieving high-quality samples for LLM prediction. Notably, even with pseudo labels, *NPL* and *NPG* outperform *GE* with true labels, suggesting that effective ICL examples can significantly improve LLM predictions, even when the labels are noisy. Finally, *NPG* outperforms *NPL* with Qwen1.5-72B, while *NPL* performs better than *NPG* with Llama3-70B. Combined with the GNN and Zero-Shot results in Table 1, we conclude that the quality of the pseudo-label is crucial for effectively using a query node's neighbors as ICL examples.

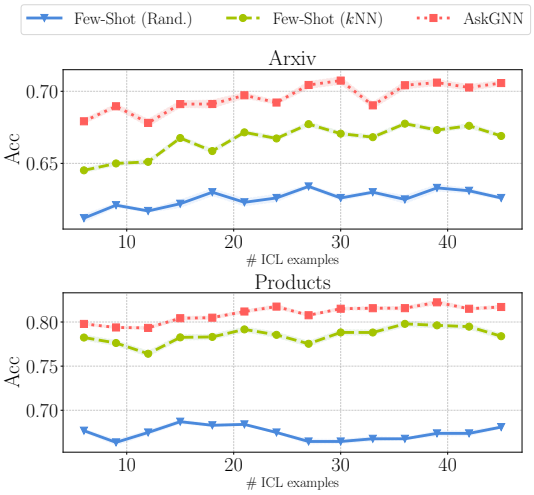

Figure 8: Performance comparison of AskGNN, Few-shot (Rand.), and Few-shot (k-NN) with varying numbers of ICL examples on the ogbn-arxiv and ogbn-products datasets.

# of ICL Examples. To evaluate the generalization ability of our approach across different numbers of ICL examples, we conducted an experiment to assess its performance under varying conditions. This analysis is essential for understanding the robustness and scalability of AskGNN across different datasets. As shown in Figure 8, we compare the sample efficiency of Few-shot (Rand.), Few-shot (k-NN) and AskGNN. We observe that performance improves for all methods as the number of ICL examples increases. This improvement is more pronounced on the ogbn-arxiv dataset than on the ogbn-products dataset. Additionally, our

method consistently outperforms Few-Shot across all datasets, demonstrating its strong generalization capability, even as the ratio of ICL examples increases.

ICL Example Purification. To address the potential impact of noise in retrieved In-Context Learning (ICL) examples, we conducted experiments exploring two noise reduction strategies through ICL example purification. Given that performance often plateaus as context length increases (Li et al., 2024a), we focused on two heuristic approaches: *(1).* LLM-Selection. This approach involves prompting the Large Language Model (LLM) to select the most informative examples from multiple retrieved instances based on criteria such as relevance and diversity. The underlying intuition is that LLMs, when tasked with curating their own input data, can identify examples that offer the most value for downstream tasks while filtering out noisy or redundant entries; *(2).* Minority-Class-Removal. This method involves removing ICL examples with less frequent labels. We hypothesized that such outliers might introduce unwanted noise into the retrieval process. By removing these minority class examples, we aimed to reduce the likelihood of including anomalous data that could detract from model performance.

The results from these experiments, conducted using Qwen1.5-72B on the obgn-arxiv dataset, are summarized in Table 5. Our findings indicate that LLM-Selection improved performance by approximately 1% (e.g., from 69.02% to 71.10% accuracy on ogbn-arxiv). In contrast, Minority-Class-Removal had a minimal impact, with changes in performance generally within ±0.1%. Overall, these additional experiments demonstrate that improving the quality of retrieved ICL examples can positively impact performance.

Table 5: Performance of ICL example purification techniques on the ogbn-arxiv dataset using Qwen1.5-72B.

	ogbn-arxiv			ogbn-products		
# of ICL Examples	30	33	36	24	27	30
AskGNN	70.74	69.02	70.43	81.44	80.76	81.50
LLM Selection	70.81	71.10	71.39	81.49	81.66	81.70
Minority-Class-Removal	70.72	69.17	70.41	81.42	80.77	81.52

6.4 Are LLMs Just Repeaters?

To determine whether LLMs merely replicate the most prevalent class from ICL examples, rather than engaging in meaningful graph reasoning and task understanding, we analyzed performance in two scenarios: Few-shot (k-NN) and AskGNN. Additionally, we implemented a Majority Voting (MV) strategy, which selects the class that appears most frequently in the ICL examples as the prediction. As shown in Table 2, MV consistently underperformed compared to LLM predictions across all experimental conditions, indicating that LLMs are making more nuanced decisions rather than simply mimicking the most common class. This outcome highlights the importance of selecting relevant examples strategically in ICL.

7 Conclusion

In this work, we introduced AskGNN, a novel approach leveraging the ICL capabilities of LLMs for graph-based tasks. By implementing a structure-enhanced retriever (SE-Retriever) based on GNNs, AskGNN bridges the gap between sequential text processing and graph-structured data while preserving the inherent supervision signals of graph nodes. Our evaluations across three distinct tasks and seven LLMs show that AskGNN significantly improves node classification performance compared to existing methods, confirming its robustness and effectiveness. Looking forward, future work will focus on extending AskGNN to a broader range of graph-based applications, such as dynamic graph analysis, where the complexity of the data increases and requires more sophisticated handling. This research establishes a promising foundation for utilizing structured graph data to expand the capabilities of LLMs in advanced applications.

8 Limitations

We also recognize the following limitations of this work: First, data quality issues, such as low-quality product descriptions, can hinder larger models more than smaller ones, affecting performance on complex datasets. Second, the limited input window of current open-source LLMs capped the number of ICL examples at 45, restricting the exploration of larger example sets that might improve performance. Last, our approach relies heavily on the learned GNN retriever, meaning inaccurate structure information could reduce overall model effectiveness.

9 Acknowledgement

This research was supported in part by NIH R01LM01372201.

References

AI@Meta. 2024. Llama 3 model card.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

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. In *NeurIPS*.

Pei Chen, Soumajyoti Sarkar, Leonard Lausen, Balasubramaniam Srinivasan, Sheng Zha, Ruihong Huang, and George Karypis. 2023. Hytrel: Hypergraph-enhanced tabular data representation learning. In *NeurIPS*.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. 2024a. Llaga: Large language and graph assistant. In *ICML*.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2024b. Exploring the potential of large language models (llms) in learning on graphs. In *SIGKDD*.

Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. 2024c. Label-free node classification on graphs with large language models (llms). In *ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Kaize Ding. 2024. Data-efficient graph learning. In *AAAI*.

Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. 2020. Be more with less: Hypergraph attention networks for inductive text classification. In *EMNLP*.

Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. 2022. Data augmentation for deep graph learning: A survey. In *SIGKDD*.

Elvis Dohmatob, Yunzhen Feng, Pu Yang, Francois Charton, and Julia Kempe. 2024. A tale of tails: Model collapse as a change of scaling laws. *arXiv preprint arXiv:2402.07043*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2023. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *ICLR*.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*.

Zhengyu Hu, Linxin Song, Jieyu Zhang, Zheyuan Xiao, Jingang Wang, Zhenyu Chen, Jieyu Zhao, and Hui Xiong. 2024. Rethinking llm-based preference evaluation. *arXiv preprint arXiv:2407.01085*.

Zhengyu Hu, Jieyu Zhang, Haonan Wang, Siwei Liu, and Shangsong Liang. 2023. Leveraging relational graph neural network for transductive model ensemble. In *SIGKDD*.

Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2023. Can llms effectively leverage graph structural information: when and why. *TMLR*.

Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy S Liang, and Jure Leskovec. 2024. Prodigy: Enabling in-context learning over graphs. In *NeurIPS*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *EMNLP*.

Bowen Jin, Chulin Xie, Jiawei Zhang, Kashob Kumar Roy, Yu Zhang, Suhang Wang, Yu Meng, and Jiawei Han. 2024a. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *ACL Finding*.

Hanlei Jin, Yang Zhang, Dan Meng, Jun Wang, and Jinghua Tan. 2024b. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024a. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023a. Unified demonstration retriever for in-context learning. In *ACL*.

Yichuan Li, Kaize Ding, and Kyumin Lee. 2023b. Grenade: Graph-centric language model for self-supervised representation learning on text-attributed graphs. In *EMNLP Findings*.

Yichuan Li, Kaize Ding, Jianling Wang, and Kyumin Lee. 2024b. Empowering large language models for textual data augmentation. In *ACL Findings*.

Yichuan Li, Xiyao Ma, Sixing Lu, Kyumin Lee, Xiaohu Liu, and Chenlei Guo. 2024c. Mend: Meta demonstration distillation for efficient and effective in-context learning. In *ICLR*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *ACL*.

Ian R McKenzie, Alexander Lyzhov, Michael Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Aaron Kirtland, Alexis Ross, Alisa Liu, et al. 2023. Inverse scaling: When bigger isn't better. *TMLR*.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2016. Rethinking the role of demonstrations: What makes in-context learning work? In *EMNLP*.

Mistralai. 2024. Mixtral-8x7b model card.

Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. 2016. Context encoders: Feature learning by inpainting. In *CVPR*.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. 2024. Let your graph do the talking: Encoding structured data for llms. *arXiv preprint arXiv:2402.05862*.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *NAACL*.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023*.

Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. 2024. Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment. In *ACL Finding*.

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2024. A survey on large language models for recommendation. In *WWW*.

Zhenyu Wu, YaoXiang Wang, Jiacheng Ye, Jiangtao Feng, Jingjing Xu, Yu Qiao, and Zhiyong Wu. 2023. Openicl: An open-source framework for in-context learning. In *ACL*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference. In *ICLR*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2022. Leveraging relational graph neural network for transductive model ensemble. In *ACL*.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *EACL*.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. Graphsaint: Graph sampling based inductive learning method. In *ICLR*.

Chuxu Zhang, Kaize Ding, Jundong Li, Xiangliang Zhang, Yanfang Ye, Nitesh V Chawla, and Huan Liu. 2022. Few-shot learning on graphs. In *IJCAI*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *ICML*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *ACL*.

A Baseline Details.

- **Bare GNN**: *GCN* (Ye et al., 2024) is a semi-supervised learning approach for graph-structured data, employing convolutional operations to aggregate and transform node features. It is particularly effective in tasks such as node classification and link prediction. *GraphSAGE* (Ye et al., 2024) is an inductive framework for learning node embeddings by sampling and aggregating features from a node's neighbors, thereby enabling scalability for large graph datasets. *GraphSAINT* (Ye et al., 2024) enhances the scalability of GNNs through graph sampling, reducing computational burden and memory usage by generating smaller subgraphs for training, making it well-suited for large-scale graph tasks.
- **Text-based Serialization**: *K-Hop* (Chen et al., 2024b) leverages 2-hop neighbors as in-context examples, utilizing the typical 2-layer structure of GNNs for enhanced context. *Graph-CoT* (Jin et al., 2024a) augments LLMs through iterative reasoning on graphs via the Graph Chain-of-Thought framework, consistently outperforming baseline methods.
- **Graph Projection**: *InstructGLM* (Ye et al., 2024) employs natural language prompts to describe graph structures, enabling generative language models to handle graph machine learning tasks. This approach eliminates the need for complex graph-specific mechanisms and supports scalable training.
- **InstructTuning**: This technique fine-tunes LLM parameters to enable the model to learn information within graph datasets (Zheng et al., 2024).
- **In-Context Learning**: *Zero-Shot* (Chen et al., 2024c) relies solely on the node's attributes for prediction without additional context. *Few-Shot (Rand.)* (Chen et al., 2024c) involves randomly selecting a few labeled examples along with the node's attributes to assist LLMs in understanding the task, utilizing random sampling for these selections. *Few-Shot (k-NN)* (Lewis et al., 2020) selects the most similar sample to the query as the in-context example based on retrieval-augmented generation.

B Prompt Design

Table 6 presents the prompts utilized across different datasets. Each prompt begins with the abstract and title of the paper, followed by a task-specific question designed to probe the model on a particular aspect of the paper and to request an explanation for the prediction. The answer section is left blank for the model to complete. Our analysis indicates that the current instructions enable the LLM to generate output that closely adheres to the expected format, with minimal deviations. This consistency facilitates the straightforward extraction of answers from the LLM's text output.

Exploring Prompt Variations. We conducted an extensive exploration of the impact of different prompts on the `ogbn-arxiv` dataset. As shown in Table 7, the performance across most prompts is generally similar. However, a slight improvement in accuracy is noted when the title is placed after the abstract. This observation supports the principle suggested by (Zhao et al., 2021), which posits that positioning more critical information later in the prompt can enhance performance.

C Dataset

We conduct experiments on three TAGs: `ogbn-arxiv`, `ogbn-products` (Hu et al., 2020), and arxiv23. Table 8 summarizes the statistics of these datasets.

C.1 Dataset Description

- **ogbn-arxiv (Hu et al., 2020).** The `ogbn-arxiv` dataset is a directed graph representing the citation network of all computer science arxiv papers indexed by MAG (Wang et al., 2020). Each node is an arxiv paper, and each directed edge represents a citation. The task is to predict the 40 subject areas of arxiv CS papers, such as cs.AI, cs.LG, and cs.OS, which are manually labeled by the authors and arxiv moderators.
- **ogbn-products (Hu et al., 2020).** The `ogbn-products` dataset represents an Amazon product co-purchasing network, with nodes representing products and edges indicating co-purchases. The task is to predict the category of a product in a multi-class classification setup, using 47 top-level categories as target labels.
- **arxiv23 (He et al., 2023).** The `arxiv23` dataset is a directed graph representing the citation network of all computer science arxiv papers published in 2023 or later. Similar to

Table 6: Prompts used in this study to query the LLM.

Dataset	Prompt
ogbn-arxiv & arxiv23	You are an AI trained to categorize arxiv computer science papers into specific categories based on their abstracts. Your task is to analyze the paper description provided and identify the most relevant category. \n Paper description:<paper description>. \n Give me the category of this content. Respond only with the category key (e.g., 'cs.AI', 'cs.SY'), without any additional text or explanation. Here are some of the papers cited by this paper: <ICL examples>: \n \n Answer:
ogbn-products	You are an AI trained to categorize products into specific categories based on their descriptions and characteristics. Your task is to analyze the product description provided, consider its characteristics, and identify the most relevant category among hundreds of possible categories. There are a total of 46 categories, including 1) Home & Kitchen, 2) Health & Personal Care, 3) Beauty, 4) Sports & Outdoors, 5) Books, 6) Patio, Lawn & Garden, 7) Toys & Games, 8) CDs & Vinyl, 9) Cell Phones & Accessories, 10) Grocery & Gourmet Food, 11) Arts, Crafts & Sewing, 12) Clothing, Shoes & Jewelry, 13) Electronics, 14) Movies & TV, 15) Software, 16) Video Games, 17) Automotive, 18) Pet Supplies, 19) Office Products, 20) Industrial & Scientific, 21) Musical Instruments, 22) Tools & Home Improvement, 23) Magazine Subscriptions, 24) Baby Products, 25) NAN, 26) Appliances, 27) Kitchen & Dining, 28) Collectibles & Fine Art, 29) All Beauty, 30) Luxury Beauty, 31) Amazon Fashion, 32) Computers, 33) All Electronics, 34) Purchase Circles, 35) MP3 Players & Accessories, 36) Gift Cards, 37) Office & School Supplies, 38) Home Improvement, 39) Camera & Photo, 40) GPS & Navigation, 41) Digital Music, 42) Car Electronics, 43) Baby, 44) Kindle Store, 45) Kindle Apps, 46) Furniture. \n Product description: <product description>. \n Consider its characteristics and give me the category of this product. Respond only with the category key (e.g., 'Electronics', 'Toys & Games'), without any additional text or explanation. \n Here are some examples to help you understand how to categorize products based on their descriptions: <ICL examples>. \n \n Answer:

ogbn-arxiv, each node is an arxiv paper, and each directed edge represents a citation. The task is to predict the 40 subject areas of arxiv CS papers, such as cs.AI, cs.LG, and cs.OS, which are manually labeled by the authors and arxiv moderators.

C.2 Dataset splits and random seeds

In our experiments, we adhered to specific dataset splits and employed random seeds for reproducibility. For the ogbn-arxiv and ogbn-products datasets, we used the standard train/validation/test split provided by OGB (Hu et al., 2020). For arxiv23 datasets, we followed the splits used in previous works (Huang et al., 2023; He et al., 2023). Additionally, we utilized 10 random seeds to ensure the reproducibility of our experiments. This approach enabled consistent evaluation of our proposed method across the respective datasets, with detailed results available in the supplementary material.

D Experiment Details

Computing Environment and Resources. The proposed method was implemented using PyG modules, which are licensed under the MIT License. Our experiments were conducted on a high-performance computing setup featuring an Intel(R) Xeon(R) Platinum 8358P CPU at 2.60GHz, with 512GB of memory. The computational resources included eight NVIDIA A6000 GPUs, each with 48GB of memory.

Hyperparameters. Table 9 presents the hyperparameters utilized for the GCN (Kipf and Welling, 2017), GraphSAGE (Hamilton et al., 2017), and GraphSAINT (Ye et al., 2024) models, as derived from the official OGB repository*. Additionally, the hyperparameters for GraphSAINT and the associated language models align with those specified in the GraphSAINT repository*. It is critical to emphasize that these hyperparameters were not individually optimized for each dataset; instead, they were uniformly applied across all three TAG datasets, in accordance with established practices. This uniform application underscores the flexibility, user-friendliness, and compatibility of our proposed method with prevailing GNN baselines.

Selection of Hyperparameter K. In our experiments, we investigated the impact of varying the hyperparameter K in Eq.4, which controls the number of examples utilized for LLM feedback. The analysis was conducted on the ogbn-products dataset using 10% of the training data. As shown in our results, setting $K = 2$ achieved an accuracy of 79.08%, while increasing K to 20 and

*https://github.com/snap-stanford/ogb
*https://github.com/GraphSAINT/GraphSAINT

Table 7: Prompts employed in our experiments to investigate the impact of various prompting strategies. The results indicate relatively uniform performance across most prompts.

Description	Prompt	Accuracy
Default prompt	You are an AI trained to categorize arxiv computer science papers into specific categories based on their abstracts. Your task is to analyze the paper description provided and identify the most relevant category. \n Paper description: Abstract: <abstract text> \n Title: <title text>. \n Give me the category of this content. Respond only with the category key (e.g., 'cs.AI', 'cs.SY'), without any additional text or explanation. Here are some of the papers cited by this paper: <ICL examples>: \n \n Answer:	0.729
Title first	You are an AI trained to categorize arxiv computer science papers into specific categories based on their titles and abstracts. Your task is to analyze the paper description provided and identify the most relevant category. \n **Title:** <title text> \n Paper description:<paper description>. \n Give me the category of this content. Respond only with the category key (e.g., 'cs.AI', 'cs.SY'), without any additional text or explanation. Here are some of the papers cited by this paper: <ICL examples>: \n \n Answer:	0.700
Focus on text content	You are an AI trained to categorize arxiv computer science papers into specific categories based on their abstracts. Your task is to analyze the paper description provided and identify the most relevant category. \n Paper description:<paper description>. \n Give me the category of this content. **Focus only on content in the actual text and avoid making false associations.** Respond only with the category key (e.g., 'cs.AI', 'cs.SY'), without any additional text or explanation. Here are some of the papers cited by this paper: <ICL examples>: \n \n Answer:	0.698
Chain of thought prompt	You are an AI trained to categorize arxiv computer science papers into specific categories based on their abstracts. Your task is to analyze the paper description provided and identify the most relevant category. \n Paper description: Abstract: <abstract text> \n Title: <title text>. \n Give me the category of this content. **Please think about the categorization in a step by step manner and avoid making false associations.** Respond only with the category key (e.g., 'cs.AI', 'cs.SY'), without any additional text or explanation. Here are some of the papers cited by this paper: <ICL examples>: \n \n Answer:	0.709

Table 8: Statistics of the TAG datasets

Dataset	#Nodes	#Edges	Task
ogbn-arxiv	169,343	1,166,243	40-class classif.
ogbn-products	2,449,029	61,859,140	47-class classif.
arxiv23	46,198	78,548	40-class-classif.

Table 9: Hyperparameters for the GCN, GraphSAGE, and GraphSAINT models.

Hyperparameters	GCN	GraphSAGE	GraphSAINT
# layers	3	3	3
hidden dim	256	256	256
learning rate	0.01	0.01	0.001
dropout	0.5	0.5	0.5
epoch	200	200	200

200 improved accuracy to 82.54% and 83.09%, respectively. However, this increase in K also led to a significant rise in computational cost, with $K = 200$ incurring approximately 10 times the cost of $K = 20$. Our findings indicate that $K = 20$ offers a balanced trade-off between accuracy and computational efficiency, providing substantial performance improvements without incurring excessive computational overhead. This flexibility in adjusting K allows the method to adapt to larger datasets or resource-constrained environments. Based on this analysis, we selected $K = 20$ as the default setting for our experiments, as it achieves an optimal balance between performance gains and computational efficiency.