Graph Learning with LLMs: An Inductive Training-free Approach

Anonymous ACL submission

Abstract

Graph Neural Networks (GNNs) exhibit state-of-the-art performance on node classification tasks, which predicts the categorical class of each node. Yet they face challenges with computational requirements and generalization to unseen nodes. Recently, Large Language Models (LLMs) have been explored for graph-related tasks yet often need to be paired with GNNs to improve the classification performance. To use an LLM directly, we propose GLINT, which addresses the existing GNN challenges by identifying relevant neighboring nodes without requiring access to the entire graph structure. Our approach is designed for limited training data settings and can be performed with low computational resources. Experiments on academic citation and e-commerce product graphs demonstrate that GLINT outperforms existing GNNs for extremely limited training data across various open-source LLM backbone models.

1 Introduction

Graphs, a data structure with graphs and edges, are used to capture relationships across various domains. One crucial task is node classification which predicts the categorical class of each node. Node classification can be used to recommend products in e-commerce platforms (Ying et al., 2018), suggest relevant papers to read (Yang et al., 2016), or even understand protein functions in biological networks (Szklarczyk et al., 2019). By aggregating information across neighboring nodes and the node features, Graph Neural Networks (GNNs) have achieved remarkable results for node classification (Wu et al., 2020). However, there remain two open challenges. First, the entire graph needs to be loaded to train the model (Wang et al., 2021). Second, GNNs may not generalize well to unseen nodes (inductive setting) (Garg et al., 2020) as existing works assume the graph structure is known at training time (i.e., predict unlabeled nodes where the edges are observed when training the model).

The increasing adoption of Large Language Models (LLMs) beyond language processing has prompted investigations into the potential of LLMs to enhance performance on graph-related tasks (Ren et al., 2024a; Jin et al., 2024). Existing research has predominantly paired LLMs with GNNs where one serves as the prefix for the other (e.g., graphs with textual features use LLMs to generate the node features or GNNs provide structure-aware tokens to LLMs for inference) or fused the LLM and GNN representations. Although such works can yield better performance for node classification, they still fail to address the two major challenges of GNNs. An alternative is to use LLMs only to perform inference for node classification. The crucial question is then how to construct the graph effectively using natural language only. There have been several works that have examined the capabilities of LLMs to understand and reason with graph data (Hu et al., 2023; Wang et al., 2024a; Tang et al., 2024; Fatemi et al., 2024; Ye et al., 2024; Tan et al., 2023; Chen et al., 2024; Zhang et al., 2024). However, these works either require additional fine-tuning (Tang et al., 2024; Ye et al., 2024; Tan et al., 2023; Chen et al., 2024) or leverage instruction tuning approaches or chain-of-thought prompting (Zhang et al., 2024; Hu et al., 2023; Wang et al., 2024a; Fatemi et al., 2024). Moreover, existing approaches require encoding the entire graph to tackle broad graph reasoning tasks such as connectivity, cycle, and shortest path. Thus large graphs will often exceed the context limit for LLMs.

In this work, we propose GLINT, a graph learning inductive training-free approach using LLMs tailed for node classification on graphs with textual features. First, we posit that LLMs are well-suited for the limited training data setting where the entire graph structure is unknown as LLMs can harness the textual features to label the node.

Second, instead of constructing the entire graph in natural language, GLINT identifies the most relevant neighboring nodes without requiring access to the entire graph. This means our approach can be performed in low computational resource settings as inference can be performed through an API call. Moreover, this provides flexibility for GLINT to generalize to LLMs with varying context lengths. Third, we propose a simple label compression to use one-shot in-context learning with multiclass tasks. Although previous works suggest few-shot learning may not yield improvements for graph-based tasks (Hu et al., 2023; Wang et al., 2021), we observe longer category text can confuse the LLMs. By using a smaller label (even without any semantic meaning), the LLM can output responses consistent with the categories, thereby removing ambiguity from the output verification process.

We verify the effectiveness of GLINT by conducting experiments on two node classification tasks: an academic citation and an e-commerce product graph. Our results show that for extremely limited training data, GLINT outperforms existing GNNS across various open-source LLM backbone models. Our analysis further confirms the advantage of leveraging relevant neighboring nodes to provide context to the LLM.

2 Methodology

2.1 Problem Setup

Given the entire original graph $G = (V, E)$, the inductive graph learning setting partitions the nodes V into disjoint subsets to assess the model's ability to generalize to new, unseen nodes or entirely new graphs without re-training. Thus, we assume 3 disjoint node subsets, V_{train}, V_{val}, and V_{test}, for training, validation, and testing, respectively. The training subgraph, $G_{train} = (V_{train}, E_{train})$, includes only edges between nodes in V_{train}, i.e., $E_{train} = \{(v_i, v_j) \in E \mid v_i, v_j \in V_{train}\}$. The validation subgraph, $G_{val} = (V_{train} \cup V_{val}, E_{val})$, contains nodes from both V_{train} and V_{val}, with edges $E_{val} = \{(v_i, v_j) \in E \mid v_i, v_j \in V_{train} \cup V_{val}\}$. This construction guarantees that G_{train} is a subgraph of G_{val}, allowing the model to apply learned structural information to unseen nodes and edges. During inference, the node classification task focuses on predicting the category of a target node $v_{target} \in V_{test}$ using the entire graph G while only V_{train} labels are available.

2.2 Selecting Relevant Neighbors

Existing LLM-only approaches encode the entire graph as textual data, which only works for small graphs. As such, GLINT utilizes a simple sampling strategy by constructing the local subgraph centered around the target node, identifying all the neighbors within 2 hops of the target node (i.e., any nodes in V within a distance of 2 from the target node). This can be done efficiently without loading the entire graph (Khandelwal et al., 2017).

To manage the potentially large number of nodes in the local subgraph, we limit the representation to only the most relevant nodes. Rather than using all nodes, we rank them by running PageRank on the local subgraph (Page, 1999), which helps identify the most influential nodes based on their connectivity. We then select the top 50 nodes to ensure the prompt remains concise without losing key information. The textual features, along with their labels (if available from V_{train}), are provided to the LLM as context for predicting the target node's label. This strategy balances subgraph size with information richness, improving the accuracy of the model's predictions.

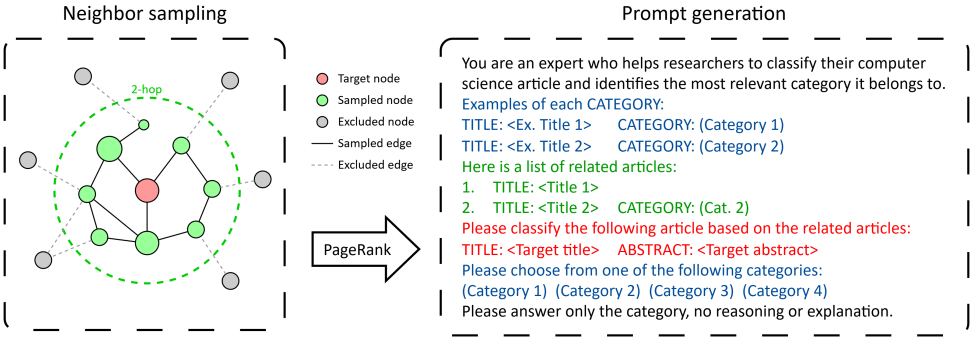
2.3 One-Shot Category Demonstrations

Given the complexity of summarizing categories in natural language, we provide the LLM with a single exemplar for each category. While previous studies suggest that few-shot demonstrations may not improve performance on graph-based tasks (Hu et al., 2023; Wang et al., 2021), we hypothesize that this could be due to unrepresentative exemplars or overly detailed category descriptions, which might confuse the LLM. To address this, we adopt PageRank again on V_{train} to rank node importance and then select the highest-ranking node from each category as its exemplar. Moreover, we streamline the prediction process by assigning each category a unique two-letter pseudo-label enclosed in parenthesis. While this removes the semantic meaning of the label, we observe that label compression leads to more consistent LLM responses by providing a clear, standardized format for predictions.

2.4 LLM Prompt Template

We then design a concise prompt concatenating the relevant neighbors, category exemplars, and pseudo-labels, structured into four parts. First, we briefly describe the persona and the node classification task. Second, we present the exemplars

Figure 1: The general process for generating the LLM prompt for GLINT. Given the local subgraph around the target node (in red), we calculate the relative importance using PageRank (node size) to determine the relevant neighbor nodes. The green text in the prompt refers to features of the sampled nodes and the red text contains information about the target node. The exemplar nodes and their associated categories are shown in blue text.



for each category by providing the textual features (e.g., paper titles in Cora) and the two-letter pseudo-label. Third, we include we include relevant neighbor nodes, with pseudo-labels for those in the training set. Finally, the prompt lists all the possible categories and requests only a category response, ensuring consistency in LLM predictions. Detailed prompt designs are available in Appendix D.

Note that our approach does not require any training and only involves additional computation during inference. The computation is relatively minimal as it only involves localized neighborhood sampling and PageRank on small subgraphs. Efficient PageRank approximations allow these operations to be performed quickly and with minimal computational cost (Mitliagkas et al., 2015; Wu et al., 2024). Furthermore, if the graph is stored in a graph database, all these computations can be executed directly within the database. LLM inference can then be performed via an API call. Thus, the local machine only needs the capability to query the graph database, construct the local graph, and run PageRank. However, LLM inference constitutes a significant portion of the overall processing time. The method's efficiency may be compromised for a large graph, where the cumulative LLM inference time may become a bottleneck.

3 Experiments

3.1 Experiment Setups

◇ **Datasets.** We tested our approach on 2 different graphs, both with associated text as features. Cora is an academic citation graph (McCallum et al., 2000) and ogbn-products is an Amazon product co-purchasing network (Hu et al., 2020). Similar to He et al. (2024), we used a subset of the original ogbn-products dataset to assess the feasibility of GLINT. Appendix B contains further dataset details.

◇ **Evaluation.** We employ 3 different settings where we vary the train/validation/test splits: (1) the original predefined train-test division (i.e., 60/20/20 for Cora and 15/15/70 split for ogbn-products); (2) a more challenging setting of 10/10/80; and (3) a tough split of 1/1/98. We assess the models on accuracy and use exact match string comparison on the LLM response.

◇ **Baselines.** We compared our proposed method by comparing with existing GNN-based methods. For GNN comparisons, we consider three widely utilized architectures that perform well in the inductive setting: (1) GraphSage (Hamilton et al., 2018) with mean, max, and LSTM aggregation functions; (2) GIN (Xu et al., 2019); and (3) GAT (Veličković et al., 2018). For GLINT, we tested 3 open-source, instruction-tuned LLMs: (1) Gemma2 27B (Team, 2024a); (2) Llama 3.1 70B (AI@Meta, 2024); and (3) Qwen 2.5 72B (Team, 2024b). See Appendix C for implementation details.

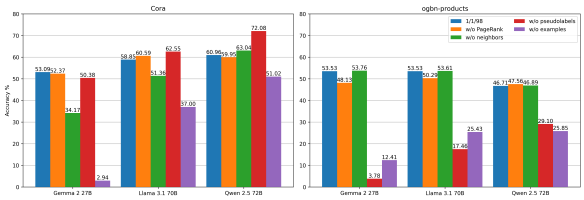
3.2 Performance Results

Table 1 summarizes the experiment results of GLINT and the GNN baselines. First, we observe GLINT across at the various LLM backbone performs comparably or better than GNNs for the extremely difficult split of 1/1/98. We posit the node features play a more significant role than the graph structures with limited training data. GNNs were more likely to overfit in this circumstance and therefore generalized poorly. Both GraphSAGE and GIN achieved perfect training accuracy on the Cora dataset with only 27 samples. GAT was less affected by the lack of training data, possibly benefiting from attending to node features in a manner similar to the LLMs. The 1/1/98 results highlight the potential promise of LLMs to utilize textual

Table 1: Results on test sets. All metrics are measured in percent accuracy. The highest accuracy for each dataset split is highlighted with **bold** font and the second highest is underlined. GNN results are from an average of 5 runs and LLM results are from a single run.

Model	cora			ogbn-products (subset)		
	original	10/10/80	1/1/98	original	10/10/80	1/1/98
GraphSAGE-LSTM (Hamilton et al., 2018)	74.32	46.96	26.04	25.73	33.80	27.76
GraphSAGE-max (Hamilton et al., 2018)	87.71	63.92	15.75	60.20	65.63	39.35
GraphSAGE-mean (Hamilton et al., 2018)	88.38	69.79	15.83	**60.64**	**67.53**	42.98
GIN (Xu et al., 2019)	87.64	61.88	16.13	59.10	64.78	48.01
GAT (Veličković et al., 2018)	**88.93**	**79.19**	56.10	53.73	63.24	47.89
Gemma-2-27B-it (Team, 2024a)	78.97	64.97	53.09	47.70	54.87	**53.53**
Llama-3.1-70B-it (AI@Meta, 2024)	83.76	71.02	58.85	47.93	53.47	**53.53**
Qwen-2.5-72B-it (Team, 2024b)	85.79	72.77	**60.96**	41.88	47.34	46.71

Figure 2: Results of the ablation studies.



features to overcome the lack of graph structure.

Second, when there is sufficient training data (original split and 10/10/80), GNNs offer better performance than LLM. We posit the model can fully integrate neighbor information to yield better node embeddings rather than solely relying on text. GLINT can only consistently outperform GraphSAGE-LSTM. We also observe the performance of all models dropped as the number of training nodes decreased, except for the original split of ogbn-products. We suspect the original subgraphs were structurally more challenging.

3.3 Ablation Study

We inspected the different components of GLINT to understand the impact of our designs. Figure 2 summarizes the results.

◇ **Effect of Relevant Neighbors.** We assess the importance of selecting the relevant nodes using PageRank compared to (i) without (w/o) PageRank which uses a random sampling procedure while maintaining the same number of neighbors to the LLMs and (ii) no neighbors. Our findings reveal that PageRank's impact varies depending on graph size and connectivity. For the smaller Cora graph where nodes have limited connections, PageRank's influence is minimal. However, in the larger, more complex network ogbn-products, PageRank yields a ∼3% improvement in accuracy. We also observe random neighbor sampling can yield worse performance than no neighbor information, suggesting that the mere presence of neighbors, without strategic selection, can impair performance.

◇ **Effect of One-Shot Category Demonstration.** We assess the effectiveness of the two-letter pseudolabels with original long labels. We find that short pseudolabels significantly enhance performance on the 47-class ogbn-products dataset, whereas the benefit is more mixed for Cora. We believe the longer labels combined with numerous classes in ogbn-products, may confuse LLMs and adversely affect their performance. Cora contains only seven classes and the shorter pseudolabels unexpectedly diminish performance for the larger-sized LLMs. This highlights the need to tailor the pseudolabel length to the task and LLM.

◇ **Effect of Few-Shot Examples.** We evaluate the importance of providing one training example for each category. Our results demonstrate that this approach significantly enhances the performance of all LLMs across both datasets. This observation suggests that LLMs benefit from explicit connections between pseudolabels and their corresponding textual contexts, highlighting the importance of grounding abstract labels in concrete examples.

4 Conclusion

We propose GLINT, a training-free LLM approach to inductive node classification. For scenarios with extremely limited training data, GLINT offers superior performance to traditional GNN approaches. The result is consistent across multiple LLMs on both Cora and ogbn-products. Our results suggest that LLMs can effectively leverage textual node features to overcome the lack of extensive structure information when training data is scarce.

Our ablation studies shed light on the key components of GLINT's efficacy. We observe using PageRank outperforms random sampling, particularly in larger networks. Our prompting strategy of using concise pseudolabels is effective on ogbn-products, while providing few-shot examples substantially improves performance on both datasets.

Limitations

In this work, we propose GLINT to leverage LLMs to perform node classification with extremely limited training data. Despite its promising performance, we note several limitations of GLINT:

◇ **Experiments on two datasets.** In this study, we only benchmarked performance comparisons on Cora and ogbn-products. There are other graph datasets from different domains with textual features that may be suitable for empirical evaluation to determine if our findings are generalizable.

◇ **Inference time.** Although this is a training-free approach that does not require extensive computational resources assuming access to an LLM API, the inference time can be quite extensive if there are a large number of nodes to label. One potential alternative to reduce this time is to investigate the performance when leveraging recent advances toward deploying and scaling LLMs for real-world deployment (Li et al., 2024).

◇ **Single graph-based Task.** In this study, we explored the direct use of LLM for node classification in the limited training setting. There are other graph-based tasks such as link prediction, subgraph classification, and graph reasoning that we have not investigated. Thus, our findings may only be relevant for node classification and not other graph-based tasks.

References

AI@Meta. 2024. Llama 3 model card.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, page 891–900, New York, NY, USA. Association for Computing Machinery.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. 2024. Llaga: Large language and graph assistant. *Preprint*, arXiv:2402.08170.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2024. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*.

Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with pytorch geometric. *Preprint*, arXiv:1903.02428.

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

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. *Preprint*, arXiv:1607.00653.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2018. Inductive representation learning on large graphs. *Preprint*, arXiv:1706.02216.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2024. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *Preprint*, arXiv:2305.19523.

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. *arXiv preprint arXiv:2005.00687*.

Yuntong Hu, Zheng Zhang, and Liang Zhao. 2023. Beyond text: A deep dive into large language models' ability on understanding graph data. *arXiv preprint arXiv:2310.04944*.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2024. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*.

Anurag Khandelwal, Zongheng Yang, Evan Ye, Rachit Agarwal, and Ion Stoica. 2017. Zipg: A memory-efficient graph store for interactive queries. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1149–1164.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *Preprint*, arXiv:1609.02907.

Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. 2024. Llm inference serving: Survey of recent advances and opportunities. *arXiv preprint arXiv:2407.12391*.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163.

Ioannis Mitliagkas, Michael Borokhovich, Alexandros G Dimakis, and Constantine Caramanis. 2015. Frogwild!–fast pagerank approximations on graph engines. *Proceedings of the VLDB Endowment*, 8(8).

Lawrence Page. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Technical Report.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024a. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6616–6626.

Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. 2024b. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, volume 36 of *KDD '24*, page 6616–6626. ACM.

Damian Szklarczyk, Annika L Gable, David Lyon, Alexander Junge, Stefan Wyder, Jaime Huerta-Cepas, Milan Simonovic, Nadezhda T Doncheva, John H Morris, Peer Bork, et al. 2019. String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, 47(D1):D607–D613.

Yanchao Tan, Zihao Zhou, Hang Lv, Weiming Liu, and Carl Yang. 2023. Walklm: A uniform language model fine-tuning framework for attributed graph embedding. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500.

Gemma Team. 2024a. Gemma.

Qwen Team. 2024b. Qwen2.5: A party of foundation models.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. *Preprint*, arXiv:1710.10903.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024a. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems*, 36.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2024b. Can language models solve graph problems in natural language? *Preprint*, arXiv:2305.10037.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2020. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *Preprint*, arXiv:1909.01315.

Zhaokang Wang, Yunpan Wang, Chunfeng Yuan, Rong Gu, and Yihua Huang. 2021. Empirical analysis of performance bottlenecks in graph neural network training and inference with gpus. *Neurocomputing*, 446:165–191.

Siyue Wu, Dingming Wu, Junyi Quan, Tsz Nam Chan, and Kezhong Lu. 2024. Efficient and accurate pagerank approximation on large graphs. *Proceedings of the ACM on Management of Data*, 2(4):1–26.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? *Preprint*, arXiv:1810.00826.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR.

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

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. 2024. Llm4dyg: Can large language models solve spatial-temporal problems on dynamic graphs? *Preprint*, arXiv:2310.17110.

A Background

◇ **Transductive learning** on graphs utilizes the entire graph structure during the training stage. This approach leverages information from both labeled and unlabeled nodes, as well as the full connectivity of the graph. Many graph learning approaches are transductive, which could not generalize to unseen data (Cao et al., 2015; Grover and Leskovec, 2016; Kipf and Welling, 2017).

◇ **Inductive learning** on graphs, in contrast, trains the model using a subgraph containing only the training nodes and their associated features and edges. It is an approach that aims to generalize from a subset of nodes to the entire graph or even to unseen graphs. The key advantage of inductive learning is its ability to make predictions on new, unseen nodes or entirely new graphs without retraining (Hamilton et al., 2018).

◇ **GraphSAGE** samples a node's neighborhood and aggregates information from the neighbors. It learns an embedding function that can generalize to unseen nodes. It also supports various aggregation functions such as mean, max, LSTM, and GCN (Hamilton et al., 2018).

◇ **Graph Isomorphism Network** (GIN) does not sample the neighbors, but rather aggregates the entire neighborhood. It also uses summation and multi-layer perceptrons (MLP) as the aggregation function (Xu et al., 2019).

◇ **Graph Attention Network** (GAT) utilizes the self-attention mechanism, which assigns different weights to different neighbors when aggregating information (Veličković et al., 2018).

◇ **PageRank** is an algorithm used to rank web pages based on their relative importance. It assigns a numerical weight to each node (web page) in a directed graph. The nodes with large number of inedges will receive higher importance scores. It employs an iterative process to propagate importance across the network and can be computed efficiently on large graphs (Page et al., 1999).

◇ **Current applications** of LLMs in graph-based tasks primarily utilize LLM responses as supplementary features to augment GNNs (Ren et al., 2024b). While direct applications of LLMs in such tasks exist, they either require additional fine-tuning (Ye et al., 2024; Tan et al., 2023; Chen et al., 2024), or encode entire graph as texts (Zhang et al., 2024; Wang et al., 2024b). Our proposed method does not involve tuning and does not explicitly specify graph structures. This strategy has allowed us to scale easily to large graphs as well as perform well on simple graphs.

B Datset Descriptions

◇ **Cora** is a well-known and widely used citation network. It consists of 2,708 computer science publications classified into 7 categories, where nodes are articles and edges represent references. The most common task is to predict the category of a paper based on its content and the citation network (McCallum et al., 2000).

◇ **ogbn-products** is part of the Open Graph Benchmark (OGB), representing an Amazon product co-purchasing network. Nodes represent products and edges between two nodes indicate the two are purchased together (Hu et al., 2020). In this work, we use a subset of the original graph split by He et al., containing 54,025 nodes.

Table 2: The statistics of the 2 graphs.

Dataset	# Nodes	# Edges	# Classes
cora	2,708	5,429	7
ogbn-products (subset)	54,025	74,420	47

Table 3: Hyperparameters for GNNs.

Model	Hidden Dim.	# Layers/Att. Heads	Dropout
GraphSAGE	384	3	0.5
GIN	384	3	0.5
GAT	384	8	0.5

C Implementation Details

All GNNs in this work are trained using PyTorch and PyTorch Geometric (PyG) (Fey and Lenssen, 2019). Table 3 lists the hyperparameters for each GNN. The models are trained with Adam optimizer with a learning rate of 0.001 and 0 weight decay.

We utilize the Deep Graph Library (DGL) for the PageRank algorithm implementation (Wang et al., 2020). Graph partitioning is accomplished through PyG's `subgraph()` utility function, while node neighbors sampling uses PyG's `NeighborLoader`.

We use LLM models from HuggingFace. For LLM performance evaluation, we enclose node labels within parentheses. An LLM's response is considered correct only if its answer within parentheses exactly matches ground truth, including all alphaneumeric and special characters. All experiments are conducted on a Linux server with 2 NVIDIA H100 GPUs. Due to memory constraints, we used int4 quantization for Llama and Qwen, and bfloat16 for Gemma.

D Prompt Design Details

We present the detailed prompt templates for the datasets as follows:

Listing 1: Prompt format for Cora.

```
You are an expert who helps researchers
to classify their computer science
articles and identifies the most
relevant category it belongs to.

Examples of each CATEGORY:
TITLE: an empirical approach...
CATEGORY: (cv)
TITLE: pruning backpropagation...
CATEGORY: (ga)
...

Here is a list of related articles:
```

```
1. TITLE: references elements...
CATEGORY: (re)
2. TITLE: learning to predict...
CATEGORY: (re)
...

Please classify the following article
based on the related articles:
TITLE: decision tree function
approximation... ABSTRACT: we present a
decision tree based approach...

Please choose from one of the following
categories: (cb) (ga) (nn) (pm) (re)
(ru) (th)

Please answer only the category, no
reasoning or explanation.
```

Listing 2: Prompt format for ogbn-products.

```
You are an expert who helps business
owners to classify their product and
identifies the most relevant category
it belongs to.

PRODUCT: gator machete with sheath
CATEGORY: (pl)
PRODUCT: multipet plush dog toy
CATEGORY: (pt)
...

Here is a list of related products:
1. PRODUCT: mutant ninja turtles...
CATEGORY: (bp)
2. PRODUCT: gillette sensor3...
...

Please classify the following product
based on the related products:
PRODUCT: aimee gowns... SUMMARY: the one
that...

Please choose from one of the following
categories: (hk) (pc) (bt) (so) (bk)
(pl) (tg) (cd) (cp) ...

Please answer only the category, no
reasoning or explanation.
```