Exploring Graph Learning Tasks with Pure LLMs: A Comprehensive Benchmark and Investigation

Yuxiang Wang[1], **Xinnan Dai**[2], **Wenqi Fan**[3], **Yao Ma**[4†]
[1]The Chinese University of Hong Kong, Shenzhen
[2]Michigan State University, USA
[3]The Hong Kong Polytechnic University, HK SAR
[4]Rensselaer Polytechnic Institute, USA
yuxiangwang1@link.cuhk.edu.cn, may13@rpi.edu

Abstract

In recent years, large language models (LLMs) have emerged as promising candidates for graph tasks. Many studies leverage natural language to describe graphs and apply LLMs for reasoning, yet most focus narrowly on performance benchmarks without fully comparing LLMs to graph learning models or exploring their broader potential. In this work, we present a comprehensive study of LLMs on graph learning tasks, evaluating both off-the-shelf and instruction-tuned models across a variety of scenarios. Beyond accuracy, we discuss data leakage concerns and computational overhead, and assess their performance under few-shot/zero-shot settings, domain transfer, structural understanding, and robustness. Our findings show that LLMs, particularly those with instruction tuning, greatly outperform traditional graph learning models in few-shot settings, exhibit strong domain transferability, and demonstrate excellent generalization and robustness. Our study highlights the broader capabilities of LLMs in graph learning and provides a foundation for future research [*].

1 Introduction

The rapid progress of large language models (LLMs), such as GPTs (Achiam et al., 2023), LLaMA (Touvron et al., 2023), Claude (Perez et al., 2022), and Deepseek (Liu et al., 2024), has revolutionized many natural language processing tasks, showcasing their ability to generalize across domains and reason with minimal supervision. Recently, researchers have begun extending LLMs to non-text domains like graphs, aiming to leverage their strong reasoning capabilities for graph tasks.

Unlike text, graphs represent structured relational data, posing new challenges for LLMs in terms of representation and reasoning. To bridge this gap, various approaches have emerged: some utilize prompt engineering to describe graph structures in natural language (Cao et al., 2024; Zhang et al., 2024b; Kim et al., 2023; Jiang et al., 2023; Wang et al., 2024a; Fatemi et al., 2023), while others integrate graph embeddings from graph neural networks (GNNs) or graph transformers (GTs) into LLMs (Chen et al., 2024b; Chai et al., 2023; Tang et al., 2024a; Perozzi et al., 2024). To further mitigate the semantic gap between graphs and text, instruction tuning (Ye et al., 2023; Tang et al., 2024a; Zhang, 2023) is introduced, enabling LLMs to better understand graph features and structures.

Meanwhile, graph learning models continue to evolve. Classic GNNs (Kipf and Welling, 2016; Hamilton et al., 2017; Veličković et al., 2017; Xu et al., 2018) rely on message passing and aggregation to capture local graph structures, but their performance often depends heavily on labeled data. To alleviate this reliance, graph self-supervised learning (SSL) methods (You et al., 2020; Velickovic et al., 2019; Hou et al., 2022) adopt a pre-training–fine-tuning paradigm, using unlabeled data to learn meaningful structural representations. In parallel, GTs (Ying et al., 2021; Zhang et al., 2020) have been proposed to overcome the locality constraints of GNNs by using self-attention to model long-range dependencies. More recently, foundational graph prompt models (Liu et al., 2023a; Huang et al., 2024a; Sun et al., 2023) have introduced the concept of graph prompts as a way to better align pre-trained models with downstream tasks, thereby enhancing generalization and adaptability.

However, existing studies on applying LLMs to graph learning tasks often adopt inconsistent experimental settings, including variations in datasets, preprocessing methods, and splitting strategies (Li et al., 2024b). These inconsistencies hinder systematic comparison and obscure a clear understanding of how LLMs truly perform relative to graph learning models. To bridge this gap, we conduct a

† Corresponding author.
[*] https://github.com/JensenYX/LLM-benchmarking

Findings of the Association for Computational Linguistics: ACL 2026, pages 7909–7942
July 2-7, 2026

comprehensive evaluation of LLMs alongside 16 diverse graph learning models, encompassing GNNs, graph SSL, GTs, LM-augmented graph models, and foundational graph prompt methods. To ensure fairness and reproducibility, we standardize data processing pipelines and splitting protocols across graph datasets, covering both node classification and link prediction tasks. Our benchmark further includes a broad spectrum of LLMs, ranging from open-source models such as Llama3B and Llama8B to proprietary systems like Qwen-max, GPT-4o, Deepseek V3, and Gemini2.5 Pro.

Our benchmarking results (see details in Section 3.2) show that pure LLMs, especially larger LLMs, perform on par with or even surpass most baseline models in node classification and link prediction tasks. Instruction tuning further boosts LLM performance, enabling even smaller models to match or exceed the performance of top baseline models. Given the promising potential of instruction tuning, we further explore how LLMs with instruction tuning perform in other critical cases.

While instruction tuning significantly enhances LLM performance, it typically relies on abundant labeled data, which is often unavailable in real-world scenarios (Xia et al., 2024). To assess its effectiveness under data scarcity, we first evaluate instruction-tuned LLMs in few-shot settings, examining whether they retain strong predictive capabilities with minimal supervision. We then explore their transferability across tasks and domains, a key property for practical deployment in low-resource environments. To further test their robustness, we introduce structural perturbations that commonly occur in real-world graphs such as missing node features, edge deletions, and reduced topological similarity. In addition, we examine potential data leakage, extend evaluation to more diverse datasets and graph classification task, and assess model performance on graph reasoning tasks such as shortest path and maximum flow. Unlike graph learning tasks (e.g., node/link/graph classification), which are probabilistic and data-driven, graph reasoning tasks are algorithmic with deterministic solutions that can be computed without learning. Evaluating both types provides a more comprehensive view of LLMs' generalization, reasoning, and robustness in realistic graph settings.

Existing Benchmarks for LLMs in Graph Learning Tasks There are some benchmarking works that explore the performance of LLMs on graph learning tasks. Studies like (Chen et al., 2024d) and (Yan et al., 2023) focus on how LLMs can enhance graph models (e.g., GNNs) rather than benchmarking pure LLMs on graph learning tasks. GraphICL (Sun et al., 2025) aims to improve LLM performance in node classification and link prediction through various graph prompts, with an emphasis on prompt engineering, but it does not explore the impact of instruction tuning on LLMs in graph learning tasks. GLBench (Li et al., 2024b) also centers on how LLMs can better assist graph models, without focusing on purely LLM-based performance in graph learning tasks. Although both (Wu et al., 2025) and (Zhu et al., 2024) consider instruction-tuned LLMs, the former mainly examines their zero-shot capabilities and integration with graph models, without addressing the broader impact of instruction tuning or its role in link prediction. The latter does not thoroughly explore the performance of instruction-tuned LLMs under few-shot/zero-shot settings, domain transfer, structural understanding, and robustness—factors that are especially critical in data-scarce scenarios. *To the best of our knowledge, our work provides one of the most comprehensive evaluations of pure LLMs on graph learning tasks incorporating instruction tuning. Moreover, we go beyond prior studies by systematically evaluating instruction tuning under practical data scarcity scenarios, providing a more thorough understanding of its impact on LLM performance in graph learning tasks.*

2 Graph Learning with Pure LLMs

In this section, we introduce how we use LLMs for important real-world graph learning tasks, including node classification and link prediction.

2.1 Prompt Design

As shown in the graph encoding part of Figure 1, we combine the original graph datasets with their corresponding raw text attributes to encode the graph into a format that LLMs can understand, i.e., prompts. The prompt formats required for node classification and link prediction differ based on the specific task.

Prompt formats for node classification Following (Huang et al., 2023), we adopt three basic prompt formats that use only the target node features, its 1-hop neighbors, or its 2-hop neighbors. In the original design, neighbor labels are included, which improves reasoning but may overly simplify the task by providing direct supervision. To better assess LLMs' ability to

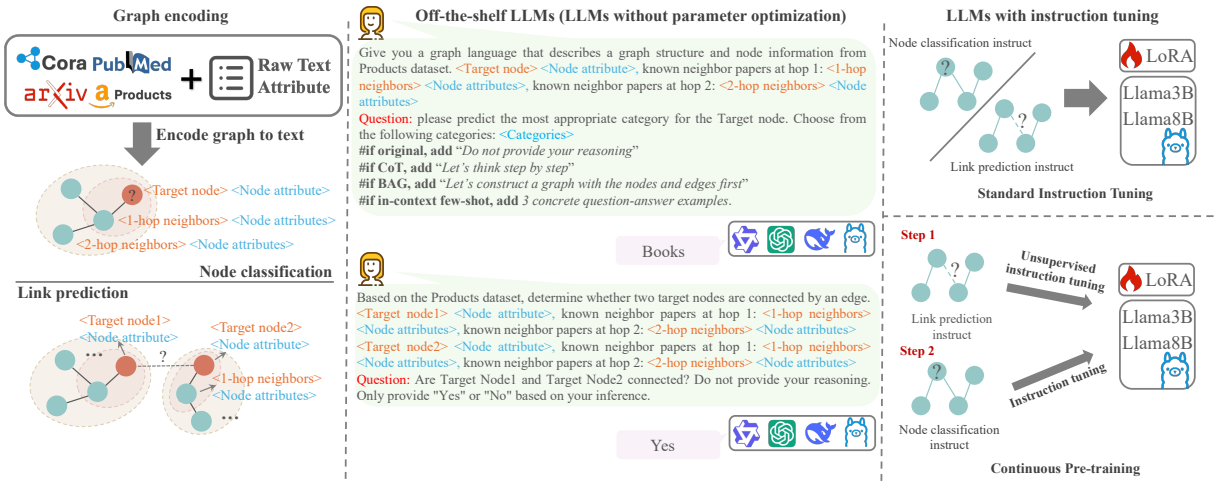

Figure 1: The overall experimental pipeline for LLMs. Graph encoding outlines how prompts for LLMs are generated. Off-the-shelf LLMs show the question-answering process with LLMs. LLMs with instruction tuning describe the process of fine-tuning LLMs specifically for graph learning tasks.

learn from structure alone, we introduce two additional formats that exclude neighbor labels. In total, we evaluate five prompt formats, and detailed prompt structures are provided in Appendix L.1:

1. **ego**: Only the target node attributes.
2. **1-hop w/o label**: Attributes of the target node and its 1-hop neighbors, excluding labels.
3. **2-hop w/o label**: Attributes of the target node and its 2-hop neighbors, excluding labels.
4. **1-hop w label**: Same as above, but with 1-hop neighbor labels from the training set.
5. **2-hop w label**: Includes 2-hop neighbor labels from the training set.

Prompt formats for link prediction We adopt two prompt formats to determine the existence of an edge between two target nodes: 1) **1-hop**: Both target nodes are described using their own node attributes and those of their 1-hop neighbors. 2) **2-hop**: extended to 2-hop neighbors. To avoid trivial cases, the two target nodes are never included in each other's neighborhood. Full prompt examples are in Appendix L.2.

2.2 Paradigm of Using LLMs for Graph Learning Tasks

As shown in Figure 1, we explore two usage paradigms: (1) off-the-shelf LLMs, which are used without parameter updates, and (2) LLMs with instruction tuning.

Off-the-shelf LLMs The LLMs we use are Llama-3.2-3B-Instruct (Llama3B), Llama-3.1-8B-Instruct (Llama8B), and the closed-source Qwen-plus (Bai et al., 2023). We directly evaluate them by feeding them carefully designed prompts encoding graph information and comparing their outputs to ground truth. Beyond basic prompts (Section 2.1), we experiment with Chain of Thought (CoT) (Wei et al., 2022), Build A Graph (BAG) (Wang et al., 2024a), and in-context few-shot prompting on larger models like Qwen-max (Bai et al., 2023), GPT-4o (Achiam et al., 2023), Deepseek V3 (Liu et al., 2024), and Gemini2.5 Pro (Comanici et al., 2025). Results show that prompt strategies vary widely in effectiveness across datasets and model scales, and do not always lead to improvements. Full comparisons are provided in Appendix J.4.

LLMs with instruction tuning We fine-tune Llama3B and Llama8B using LoRA (Hu et al., 2021a), with one training epoch per model, as longer training shows limited gains. For node classification, tuning is limited to the ego, 1-hop w/o label, and 2-hop w/o label formats. For link prediction, we examine both the benefits of tuning and the role of prompt diversity, an aspect underexplored in prior work. Two tuning modes are used: one aligns with the test formats (1-hop, 2-hop), and the other introduces nine diverse formats varying in question style and neighbor scope. Full prompt details are in Appendix L.2.

3 Comprehensive Benchmarking of LLMs for Graph Learning Tasks

Existing studies on LLMs for graphs (Tang et al., 2024a; Zhao et al., 2023; Li et al., 2024b) vary in datasets, preprocessing, and splitting, making comparisons difficult and obscuring LLM performance. Many works also benchmark against only limited baselines: (Yan et al., 2023; Ye et al., 2023)

focus on classic GNNs and GTs, while (Tang et al., 2024a) considers only GNNs and graph SSL models, overlooking recent foundational graph prompt models (e.g., OFA (Liu et al., 2023a)). This narrow scope limits insight into LLM strengths and weaknesses in graph learning. Thus, we build a comprehensive benchmark covering a broader range of models for node classification and link prediction.

3.1 The Overall Setup

3.1.1 Baselines

For baseline models, we conduct a comprehensive comparison across 6 graph learning paradigms, covering a total of 16 graph models, including both traditional GNNs and more advanced architectures. This ensures a thorough evaluation of the capabilities of LLMs. The details about baseline models can be found in Appendix G.

3.1.2 Datasets

For both node classification and link prediction, we use the Cora (McCallum et al., 2000), PubMed (Sen et al., 2008), OGBN-ArXiv (Hu et al., 2020), and OGBN-Products (Hu et al., 2020) datasets. For baseline models, we use their original node features (Appendix D discusses the impact of different node feature embedding methods). For LLMs, we preprocess the raw data to transform the node attributes into textual representations. Detailed descriptions of the datasets and their splitting methods can be found in Appendix C.

3.1.3 Evaluation Settings

For both node classification and link prediction, we consistently use accuracy as the evaluation metric, the same as (Chen et al., 2024b) and (Ye et al., 2023). In the case of link prediction, where the ratio of positive to negative samples in the test set is 1:1, accuracy is a suitable measure. To select the best model, we perform hyperparameter tuning, as different hyperparameters may cause model performance to vary across datasets. Detailed experimental settings and the hyperparameter search ranges for each model are provided in Appendix E.

3.2 Results and Analysis

We present and analyze the performance of various models across node classification and link prediction tasks, providing insights into the strengths and weaknesses of LLMs.

Node classification Table 1 summarizes the performance across different datasets. We make some observations:

- Classic GNNs show consistent accuracy, while GIANT (Chien et al., 2021) and TAPE (He et al., 2023) outperform them by using language models for improved node representations. Multiple-hop prompts yield better results than simpler prompts, indicating that LLMs benefit from richer graph context.
- Label information improves performance by strengthening the model decision-making process, similar to in-context learning.
- For instruction-tuned LLMs, Llama3B/8B shows notable improvements, especially with multiple-hop prompts. Tuned Llama8B achieves the highest average score, surpassing LLaGA (Chen et al., 2024b) and setting a new benchmark.

Model	Prompt	Cora	PubMed	ArXiv	Products	Avg
GCN	-	88.19	88.00	69.90	82.30	82.10
GraphSAGE	-	**89.67**	89.02	71.35	82.89	83.23
GAT	-	88.38	87.90	68.69	82.10	81.77
GraphCL	-	83.58	82.86	67.87	80.20	78.63
GraphMAE	-	75.98	82.82	65.54	77.32	75.42
Graphormer	-	81.20	88.05	71.99	81.75	80.75
Prodigy	-	77.32	83.6	70.86	80.01	77.95
OFA	-	78.31	78.56	73.92	83.12	78.48
GIANT	-	89.10	90.48	74.41	**84.33**	84.58
TAPE	-	88.12	91.92	73.99	83.11	84.29
LLaGA	-	88.94	**94.57**	**76.25**	83.98	**85.94**
Llama3B	ego	24.72	63.20	23.10	40.80	37.96
	1-hop w/o label	39.48	64.50	29.50	53.00	46.62
	2-hop w/o label	49.63	69.90	29.50	56.10	51.28
	1-hop w label	77.49	70.90	**66.00**	68.80	70.80
	2-hop w label	**83.03**	**72.00**	65.20	**71.20**	**72.86**
Llama8B	ego	43.39	77.80	59.35	50.12	57.92
	1-hop w/o label	58.35	73.07	61.85	59.85	63.28
	2-hop w/o label	62.84	**83.29**	**68.33**	59.60	68.52
	1-hop w label	82.97	81.55	68.08	71.07	75.92
	2-hop w label	**84.79**	82.54	64.09	**77.06**	**77.12**
tuned Llama3B	ego	67.08	89.28	66.58	65.59	72.13
	1-hop w/o label	82.04	90.02	71.32	73.07	79.11
	2-hop w/o label	**85.04**	**91.52**	**72.82**	**77.89**	**81.82**
tuned Llama8B	ego	77.31	92.36	65.59	73.74	77.25
	1-hop w/o label	84.54	93.90	69.33	80.33	82.03
	2-hop w/o label	**89.67**	**95.22**	**76.01**	**84.51**	**86.35**

Table 1: Performance of different models on node classification. The **best** results in each category are highlighted. The underline means the overall best result.

Link prediction The results for link prediction are presented in Table 2. We make the following observations:

- For baseline models, OFA (Liu et al., 2023a) achieves the best, benefiting from LLM-derived edge features during pre-training.
- Off-the-shelf Llama3B/8B lag behind most baselines, while the larger Qwen-plus matches or surpasses them, underscoring the importance of model scale for graph reasoning.
- Instruction-tuned LLMs achieve the best link prediction results. Using 2-hop prompts consistently outperforms 1-hop prompts, and tuning with 9 diverse formats yields better performance than with only 2, highlighting the value of rich structural prompts for reasoning.

Remark 1. *Although smaller off-the-shelf LLMs underperform most baseline models, their reasoning ability improves as the model size increases and graph structure information is incorporated. Instruction tuning further enhances LLM performance on graph learning tasks, with even smaller models achieving performance comparable to or better than the best baseline models, particularly when more diverse instructions are applied.*

Models	Prompts	Cora	PubMed	ArXiv	Products	Avg
GCN	-	87.78	86.22	90.34	89.75	88.52
GraphSAGE	-	84.39	78.81	92.98	92.98	87.29
GAT	-	86.88	82.81	83.33	85.57	84.65
GraphCL	-	92.98	93.76	90.85	94.21	92.95
GraphMAE	-	82.01	75.71	85.24	88.32	82.82
Prodigy	-	90.9	91.67	89.22	92.99	91.2
OFA	-	94.19	98.05	95.84	96.90	96.25
LLaGA	-	87.01	90.10	93.88	95.67	91.67
Llama3B	2-hop	68.21	59.95	68.55	79.17	68.97
Llama8B	2-hop	89.39	77.30	92.30	90.77	87.44
Qwen-plus	2-hop	90.91	95.04	93.39	90.12	92.37
t-Llama3B (2 formats)	1-hop	83.12	93.95	92.20	90.07	89.84
	2-hop	<u>95.76</u>	98.35	95.45	94.65	96.05
t-Llama3B (9 formats)	1-hop	87.18	94.40	93.30	95.45	92.58
	2-hop	**95.94**	**99.20**	95.42	<u>97.84</u>	**97.10**
t-Llama8B (2 formats)	1-hop	88.65	95.12	93.65	93.23	92.66
	2-hop	95.39	98.77	**96.11**	94.92	96.30
t-Llama8B (9 formats)	1-hop	88.47	96.01	95.21	96.33	94.01
	2-hop	95.15	**99.20**	<u>95.89</u>	**97.98**	<u>97.06</u>

Table 2: LLM performance on link prediction. The **best** and <u>second-best</u> are highlighted. t-Llama3B means tuned Llama3B, t-Llama8B means tuned Llama8B.

A theoretical justification for the effectiveness of instruction tuning is provided in Appendix H.

4 Further Investigation on LLMs with Instruction Tuning

Instruction tuning enables even small LLMs to perform well, but data scarcity remains a major challenge in real-world scenarios (Xia et al., 2024). Traditional graph models like GNNs and graph transformers often suffer under limited labeled data due to their reliance on structural and label information (Yu et al., 2024). Recent models such as All in One (Sun et al., 2023) and GPF-plus (Fang et al., 2024) aim to improve performance in low-label settings, yet the behavior of instruction-tuned LLMs under such constraints is still underexplored. Therefore, in this section, we discuss methods to alleviate data scarcity and further explore the performance of LLMs with instruction tuning in such scenarios.

Label scarcity is a common data limitation. Improving few-shot performance remains a key goal for both graph models (Yu et al., 2024; Zhao et al., 2024) and LLMs. For LLMs, few-shot instruction tuning sheds light on their robustness to label scarcity and their ability to generalize from limited supervision—crucial for real-world applicability. This motivates the following research question:

> ***RQ1: How well do LLMs perform in few-shot instruction tuning scenarios?***

When labeled data is limited, leveraging unlabeled data is a natural way to improve model performance. This idea underlies continual learning, where models incrementally adapt to new data with minimal supervision (Wang et al., 2024c; Van de Ven and Tolias, 2019). For LLMs, continual domain-adaptive pre-training (Ke et al., 2023; Yıldız et al., 2024) has proven effective for enhancing downstream performance. Inspired by this, we propose continuous pre-training for graph tasks, where LLMs are first unsupervisedly trained on graph-structured data, then fine-tuned with task-specific data. As unlabeled graph data is far more abundant, this approach holds promise for improving LLM adaptability. This motivates the following research question:

> ***RQ2: How does continuous pre-training impact the performance of LLMs?***

Models with strong transferability can mitigate performance drops under label scarcity by transferring knowledge from other datasets. LLMs have shown impressive transferability in natural language tasks (Du et al., 2024; Ran et al., 2024), but their transferability in graph tasks has been less explored. If instruction-tuned LLMs can generalize well across different graph domains, a one-time tuning process could support multiple downstream tasks, greatly reducing resource costs. This raises the following research question:

> ***RQ3: How well do LLMs transfer knowledge across domains in node classification and link prediction?***

Further Probing Given the central role of structural information in graph learning tasks, along with the practical challenges posed by real-world perturbations (e.g., missing edges) and the high computational cost of LLMs, we further investigate three key aspects: structural understanding, robustness to structural noise, and computational efficiency. Detailed analyses are provided in Appendix J.1, J.2, and J.3.

5 Experiment and Analysis

We conduct empirical studies on different research questions proposed in Section 4. In the following

Models	Prompts	Full fine-tune					5-shot					10-shot				
		Cora	PubMed	ArXiv	Products	Avg	Cora	PubMed	ArXiv	Products	Avg	Cora	PubMed	ArXiv	Products	Avg
GCN	-	88.19	88.00	69.90	82.30	82.10	62.13	68.19	24.62	47.77	50.68	71.75	71.81	25.63	54.60	55.95
GraphSAGE	-	**89.67**	**89.02**	**71.35**	**82.89**	**83.23**	58.91	65.58	19.12	45.94	47.39	70.29	70.90	22.91	51.29	53.85
GAT	-	88.38	87.90	68.69	82.10	81.77	54.95	63.95	19.08	32.65	42.66	69.26	70.60	25.34	43.59	52.20
GraphCL	-	83.58	82.86	67.87	80.20	78.63	54.03	54.86	11.24	34.10	38.56	57.96	55.23	16.84	46.08	44.03
GraphMAE	-	75.98	82.82	65.54	77.32	75.42	24.44	**70.47**	24.26	50.61	42.45	30.59	**73.63**	28.64	57.55	47.60
All in one	-	-	-	-	-	-	50.98	60.49	16.34	41.18	42.25	51.66	61.93	20.42	47.73	45.44
GPF-plus	-	-	-	-	-	-	**67.00**	66.91	60.07	**64.50**	64.62	**73.22**	64.39	65.35	**68.02**	67.75
GraphPrompt	-	-	-	-	-	-	65.12	68.11	**81.88**	58.44	**68.39**	69.81	70.38	**87.05**	61.02	**72.07**
Llama3B	2-hop w/o label	85.04	91.52	72.82	77.89	81.82	76.81	71.32	55.24	67.32	67.67	77.81	85.53	63.33	68.11	73.70
Llama8B	2-hop w/o label	**89.67**	**95.22**	**76.01**	**84.51**	**86.35**	**77.10**	**79.43**	**69.78**	**73.12**	**74.86**	**80.55**	**88.89**	**71.12**	**74.86**	**78.86**

Table 3: Performance of models under few-shot learning. The **best** results in each category are highlighted. The underline means the overall best result.

subsections, we first introduce the experimental settings for each RQ, followed by experimental results analysis and key remarks.

5.1 Few-Shot Instruction Tuning (RQ1)

5.1.1 Experiment Settings

We focus on few-shot instruction tuning for node classification. We use 2-hop w/o label as prompt formats and randomly select 5 or 10 target nodes per class for instruction tuning. For baselines, we include not only GNNs and graph SSL models but also foundational prompt-based methods including All in One (Sun et al., 2023), GPF-plus (Fang et al., 2024), and GraphPrompt (Liu et al., 2023b), which leverage pre-trained knowledge and graph prompts to perform well in few-shot settings.

5.1.2 Results

Table 3 summarizes the results. All models experience a decline in accuracy under few-shot learning compared to full fine-tuning, with GNNs and Graph SSL models showing the largest drops, particularly in larger datasets like ArXiv and Products. In contrast, LLMs exhibit more consistent performance, indicating greater robustness in data-scarce scenarios. Notably, Llama8B achieves the highest classification accuracy in both 5-shot and 10-shot scenarios, showing LLMs' ability to learn quickly from limited data.

Remark 2. *LLMs outperform all other models in few-shot scenarios. Only a few foundational graph prompt models achieve comparable results on certain datasets, underscoring LLMs' clear advantage in data-scarce situations.*

5.2 Impact of Continuous Pre-training (RQ2)

As shown in Figure 1, continuous pre-training involves two stages. First, the model undergoes task-agnostic unsupervised learning on the target dataset to acquire general graph representations. It is then instruction-tuned on a task aligned with the inference objective.

5.2.1 Experiment Settings

We evaluate both zero-shot and few-shot node classification. In the zero-shot setting, the model is first continuously pre-trained via unsupervised link prediction, then directly evaluated on node classification. Baselines include LLaGA and ZeroG (Li et al., 2024a), a prompt-based model tailored for zero-shot tasks. In the few-shot setting, we apply instruction tuning either with or without the preceding link prediction step for comparison.

Models	Prompts	Cora	PubMed	ArXiv	Products	Avg
ZeroG	-	68.61	78.77	70.50	55.23	68.28
LLaGA	-	22.03	55.92	21.15	38.90	34.50
Llama3B	2-hop	49.63	69.90	29.50	56.10	51.28
Llama3B w CPT	2-hop	55.36	75.56	33.54	57.01	55.37
Llama3B w 5s	2-hop	76.81	71.32	55.24	67.32	67.67
Llama3B w CPT & 5s	2-hop	**79.58**	88.53	54.11	68.08	72.58
Llama8B	2-hop	62.84	83.29	68.33	59.60	68.52
Llama8B w CPT	2-hop	70.82	86.96	**71.34**	63.20	73.08
Llama8B w 5s	2-hop	77.1 0	79.43	69.78	73.12	74.86
Llama8B w CPT & 5s	2-hop	78.12	**89.03**	71.01	**74.69**	**78.21**

Table 4: Performance of continuous pre-training for LLM. "w CPT" means zero-shot inference after continuous pre-training. "w 5s" means direct 5-shot instruction tuning without continuous pre-training. "w CPT & 5s" means 5-shot instruction tuning after continuous pre-training. The **best** and second-best are highlighted.

5.2.2 Results

Table 4 shows that continuous pre-training improves LLM performance over zero-shot (e.g., ZeroG) and few-shot learning, highlighting its effectiveness in enhancing graph understanding. On smaller datasets like Cora and PubMed, Llama3B with continuous pre-training matches Llama8B. However, on larger datasets such as Arxiv and Products, LLaMA-8B still leads, suggesting that model scaling remains crucial for complex graphs.

Remark 3. *Continuous pre-training can significantly improve LLM performance in zero-shot and few-shot learning. However, for larger and more complex datasets, increasing the size of the LLM proves to be a more effective approach.*

5.3 Domain Transferability of LLMs (RQ3)

Domain transferability can be divided into in-domain (across datasets within the same domain) and cross-domain (across different domains) based on difficulty. We evaluate LLMs with instruction tuning in both settings.

5.3.1 Experiment Settings

In the in-domain setting, we train on the Arxiv citation graph and evaluate on Cora, another citation dataset. For cross-domain transfer, we train on Arxiv and test on Products, an e-commerce graph. GNNs rely on task-specific heads, limiting their zero-shot capability when label sets differ, so we focus on LLaGA for node classification. For link prediction, we apply a simple linear mapping to align feature dimensions across datasets. Baselines include GNNs, graph SSL models, and LLaGA.

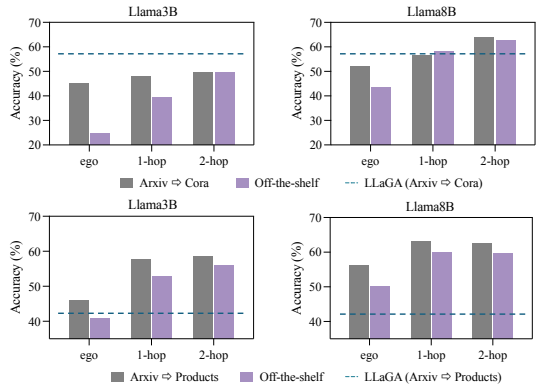

Figure 2: LLM domain transferability in node classification

5.3.2 Results

Node classification Figure 2 presents the accuracy of different models in both in-domain and cross-domain scenarios. Instruction-tuned LLMs on Arxiv outperform off-the-shelf scenario, but the improvement is modest when additional structural information is incorporated. This is likely due to the fact that node classification relies heavily on category information, and adding more structural data does not significantly enhance performance. While LLMs learn graph information from Arxiv, adapting to unseen categories remains challenging, limiting performance gains. Besides, LLMs perform comparably to LLaGA on Cora dataset, but on the more complex Products dataset, LLMs show a clear advantage. This suggests that the simple graph projector of LLaGA struggles to capture diverse graph patterns, while LLMs can adapt better to varying structures and are capable of learning diverse feature information with their sophisticated instruction tuning mechanisms.

Models	Prompts	Train ⟶ Test: Arxiv ⟶ Cora	Arxiv ⟶ Products
GCN	-	55.54	67.07
GraphSAGE	-	50.00	51.11
GAT	-	85.41	71.18
GraphCL	-	78.30	82.62
GraphMAE	-	71.90	73.94
LLaGA	-	86.98	92.82
Llama3B	1-hop	87.55	91.16
	2-hop	**95.11**	<u>94.15</u>
Llama8B	1-hop	88.98	91.97
	2-hop	<u>94.78</u>	**95.43**

Table 5: LLM domain transferability in link prediction. The **best** and <u>second-best</u> are highlighted.

Link prediction From Table 5, we observe that LLMs significantly outperform traditional graph models. Only LLaGA achieve comparable performance, likely because it also leverages LLMs for predictions. In the in-domain transfer scenario, LLMs achieve performance on Cora comparable to models directly instruction-tuned on Cora, indicating they can effectively transfer knowledge from larger datasets to downstream tasks. In the cross-domain scenario, although LLM performance on Products is slightly lower than direct tuning, it still remains strong, possibly due to shared topological patterns across domains.

Remark 4. *LLMs with instruction tuning exhibit strong domain transferability, particularly in link prediction tasks, where they effectively generalize across different datasets. This may be because link prediction tasks across domains share more similarities, as they can be viewed as binary classification problems. In contrast, node classification is more challenging, as adapting learned knowledge to unseen categories is difficult.*

6 Further Probing

To deepen our analysis, we extend evaluation to more diverse datasets and graph classification task, examine potential data leakage during pretraining, and assess model performance on graph reasoning tasks such as shortest path.

6.1 Broader Dataset and Task Evaluation

To strengthen the reliability of our conclusions, we conduct additional node classification experiments on four datasets and further evaluate graph classification tasks. As shown in Table 6, the results remain consistent with Section 3: incorporating graph structure significantly improves LLM performance, and instruction tuning further enhances their effectiveness on graph learning tasks.

Model	Prompt	Computer	WikiCS	Reddit	Instagram	IMDB-B	HIV
Task type	node	node	node	node	node	graph	graph
GCN	-	88.28	82.95	65.31	64.32	74.01	75.12
GraphCL	-	76.52	84.84	62.19	63.10	71.16	77.36
Graphormer	-	77.61	**86.17**	66.30	62.32	73.28	79.62
OFA	-	87.70	78.52	63.91	61.70	76.38	77.99
TAPE	-	89.70	83.60	63.97	65.11	-	-
LLaGA	-	**90.11**	80.01	**67.10**	66.60	-	-
Llama8B	ego	55.72	39.72	42.10	39.02	-	-
	2-hop	67.00	70.31	51.92	46.60	-	-
	-	-	-	-	-	68.51	67.12
t-Llama8B	ego	69.10	73.82	58.19	57.03	-	-
	2-hop	86.40	85.31	66.62	**68.31**	-	-
	-	-	-	-	-	**83.01**	**85.13**

Table 6: Node classification of different models on more datasets. The **best** and second-best are highlighted.

6.2 Data Leakage Analysis

A key concern is the benchmark datasets may have been seen during LLM pre-training, posing a risk of data leakage. Following (Huang et al., 2023), we compare ogbn-arxiv with arxiv-2023, a newly collected dataset of CS papers from year 2023. The two share similar citation structures, with aligned in-/out-degree distributions. We evaluate LLaMA-2-13B (trained on data up to September 2022) (Touvron et al., 2023) on node classification over both datasets, using the same setup as in Section 3.

Table 7 presents the results. If data leakage were a key factor, LLMs would show a larger accuracy drop on the leakage-free arxiv-2023 than baseline models. However, the accuracy gap is comparable across models, and in some cases, LLMs even perform better on arxiv-2023. Graph structure and instruction tuning remain the dominant contributors to performance gains. These results suggest that any potential leakage has minimal impact, and that LLMs generalize well across datasets with different temporal distributions.

Model	Prompt	ogbn-arxiv	arxiv-2023
GCN	-	69.90	65.33
GraphCL	-	67.87	66.82
Graphormer	-	**71.99**	**69.08**
Llama13B	ego	55.24	57.70
	1-hop w/o label	59.03	59.42
	2-hop w/o label	65.90	63.02
tuned Llama13B	ego	66.17	65.20
	1-hop w/o label	75.45	**76.01**
	2-hop w/o label	**76.51**	75.82

Table 7: Performance of different models on node classification tasks. The datasets are ogbn-arxiv and arxiv-2023. The **best** results in each category are highlighted. The underline means the overall best result.

6.3 Graph Reasoning Task Evaluation

While our main focus is benchmarking LLMs on data-driven graph learning tasks such as node classification and link prediction—which involve learning from data, handling uncertainty, and generalizing to unseen structures—LLMs can also be applied to graph reasoning tasks like shortest path and maximum flow. Unlike learning tasks, these problems have deterministic solutions via classical algorithms and require no training. Prior work (Wang et al., 2024a; Zhang et al., 2024a; Dai et al., 2024a; Luo et al., 2024a) has examined LLMs in this setting; here, we further study the effect of instruction tuning. We generate random graphs with 10–20 nodes using NetworkX (Hagberg et al., 2008) and use the NLGraph (Wang et al., 2024a) prompt format to evaluate LLMs on 4 tasks: connectivity, circle, shortest path, and maximum flow.

Model	Connectivity	Cycle	Shortest path	Maximum flow
Random	50	50	6.22	2.54
Llama3B	56.48	42.37	2.52	2.49
Llama8B	63.97	53.89	11.24	4.01
Qwen-max	72.85	67.27	30.07	11.90
GPT-4o	78.25	66.75	41.69	13.72
Gemini2.5 Pro	83.13	73.15	39.22	**16.38**
tuned Llama3B	74.73	79.77	25.58	11.85
tuned Llama8B	**86.40**	**88.42**	**56.98**	13.47

Table 8: Performance of different models on graph reasoning tasks. The **best** results in each category are highlighted. The underline means the overall best result.

Table 8 shows that closed-source LLMs perform reasonably well on graph reasoning tasks, and instruction tuning further improves their understanding—consistent with observations from graph learning tasks. However, for reasoning problems, classical algorithms (e.g., Dijkstra (Dijkstra, 2022) for shortest path) remain more effective and reliable. In contrast, applying LLMs to graph learning tasks is more practical and meaningful in real-world settings.

7 Conclusion

This paper demonstrates that LLMs, especially with instruction tuning, achieve strong performance and surpass most graph models on graph learning tasks through a fair and comprehensive benchmarking approach. Our findings emphasize the potential of LLMs in few-shot learning, transferability, and understanding graph structures in data-scarce scenarios. The introduction of continuous pre-training further boosts LLM performance in such environments. These insights provide valuable guidance for the future application of LLMs in graph tasks, paving the way for more efficient and adaptable graph learning models in real-world settings.

8 Limitations

Our work, while comprehensive, has certain limitations that open avenues for future research. Firstly, our investigation primarily centers on node-level and link-level predictive tasks using text-attributed graphs, where LLMs can naturally leverage their semantic processing capabilities. The performance of pure LLM approaches on graph-level tasks (e.g., graph classification and regression) or on graphs with non-semantic, numerical, or sparse features is less explored. The generalization of our findings to these contexts remains an important open question.

Secondly, scalability presents a practical challenge. The LLM-based methods, particularly those involving instruction tuning, incur significantly higher computational costs than traditional GNNs. Furthermore, the approach of serializing graph neighborhoods into textual prompts is inherently constrained by the finite context window of LLMs. This may pose difficulties when applied to graphs with extremely large or dense neighborhoods, highlighting a need for future work on more efficient graph-to-text representations or sampling strategies.

9 Ethical Considerations

The datasets utilized in our study, such as Cora and PubMed, are publicly available benchmarks for academic research, minimizing direct ethical risks. However, we acknowledge broader ethical implications inherent in applying LLMs to graph data. Potential risks include the inference of sensitive user information from graph structures and the amplification of societal biases present in the textual data. Additionally, the significant computational resources required for training and fine-tuning these models contribute to a considerable environmental footprint. Future research should prioritize the development of privacy-preserving techniques and more computationally efficient models to mitigate these concerns.

References

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Yukun Cao, Shuo Han, Zengyi Gao, Zezhong Ding, Xike Xie, and S Kevin Zhou. 2024. Graphinsight: Unlocking insights in large language models for graph structure understanding. *arXiv preprint arXiv:2409.03258*.

Ziwei Chai, Tianjie Zhang, Liang Wu, Kaiqiao Han, Xiaohai Hu, Xuanwen Huang, and Yang Yang. 2023. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*.

Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. 2024a. Graphwiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 353–364.

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

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and 1 others. 2024c. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61.

Zhikai Chen, Haitao Mao, Jingzhe Liu, Yu Song, Bingheng Li, Wei Jin, Bahare Fatemi, Anton Tsitsulin, Bryan Perozzi, Hui Liu, and 1 others. 2024d. Text-space graph foundation models: Comprehensive benchmarks and new insights. *arXiv preprint arXiv:2406.10727*.

Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. 2023. Label-free node classification on graphs with large language models (llms). *arXiv preprint arXiv:2310.04668*.

Yao Cheng, Yige Zhao, Jianxiang Yu, and Xiang Li. 2024. Boosting graph foundation model from structural perspective. *arXiv preprint arXiv:2407.19941*.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266.

Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. 2021. Node feature extraction by

self-supervised multi-scale neighborhood prediction. *arXiv preprint arXiv:2111.00064*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Xinnan Dai, Haohao Qu, Yifen Shen, Bohang Zhang, Qihao Wen, Wenqi Fan, Dongsheng Li, Jiliang Tang, and Caihua Shan. 2024a. How do large language models understand graph patterns? a benchmark for graph pattern comprehension. *arXiv preprint arXiv:2410.05298*.

Xinnan Dai, Qihao Wen, Yifei Shen, Hongzhi Wen, Dongsheng Li, Jiliang Tang, and Caihua Shan. 2024b. Revisiting the graph reasoning ability of large language models: Case studies in translation, connectivity and shortest path. *arXiv preprint arXiv:2408.09529*.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Edsger W Dijkstra. 2022. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: his life, work, and legacy*, pages 287–290.

Wenyu Du, Shuang Cheng, Tongxu Luo, Zihan Qiu, Zeyu Huang, Ka Chun Cheung, Reynold Cheng, and Jie Fu. 2024. Unlocking continual learning abilities in language models. *arXiv preprint arXiv:2406.17245*.

Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. 2024. Universal prompt tuning for graph neural networks. *Advances in Neural Information Processing Systems*, 36.

Yi Fang, Bowen Jin, Jiacheng Shen, Sirui Ding, Qiaoyu Tan, and Jiawei Han. 2025. Graphgpt-o: Synergistic multimodal comprehension and generation on graphs. *Preprint*, arXiv:2502.11925.

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

Aric Hagberg, Pieter J Swart, and Daniel A Schult. 2008. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States).

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Bryan Hooi, and 1 others. 2023. Explanations as features: Llm-based features for text-attributed graphs. *arXiv preprint arXiv:2305.19523*, 2(4):8.

Yufei He and Bryan Hooi. 2024. Unigraph: Learning a cross-domain graph foundation model from natural language. *arXiv preprint arXiv:2402.13630*.

Zhenyu Hou, Haozhan Li, Yukuo Cen, Jie Tang, and Yuxiao Dong. 2024. Graphalign: Pretraining one graph neural network on multiple graphs via feature alignment. *arXiv preprint arXiv:2406.02953*.

Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021a. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

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. *Advances in neural information processing systems*, 33:22118–22133.

Yang Hu, Haoxuan You, Zhecan Wang, Zhicheng Wang, Erjin Zhou, and Yue Gao. 2021b. Graph-mlp: Node classification without message passing in graph. *arXiv preprint arXiv:2106.04051*.

Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2023. Can llms effectively leverage graph structural information: when and why. *arXiv preprint arXiv:2309.16595*.

Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy S Liang, and Jure Leskovec. 2024a. Prodigy: Enabling in-context learning over graphs. *Advances in Neural Information Processing Systems*, 36.

Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. 2024b. Can gnn be good adapter for llms? In *Proceedings of the ACM Web Conference 2024*, pages 893–904.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pre-training of language models. *arXiv preprint arXiv:2302.03241*.

Shima Khoshraftar, Niaz Abedini, and Amir Hajian. 2025. Graphit: Efficient node classification on text-attributed graphs with prompt optimized llms. *arXiv preprint arXiv:2502.10522*.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. 2023. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models. *arXiv preprint arXiv:2310.11220*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. 2024a. Zerog: Investigating cross-dataset zero-shot transferability in graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1725–1735.

Yuhan Li, Peisong Wang, Xiao Zhu, Aochuan Chen, Haiyun Jiang, Deng Cai, Victor Wai Kin Chan, and Jia Li. 2024b. Glbench: A comprehensive benchmark for graph with large language models. *arXiv preprint arXiv:2407.07457*.

Chanuk Lim, Kyong-Ha Lee, Hyun Ji Jeong, and Sungsu Lim. 2025. Grail: Graph retrieval-augmented in-context learning for node classification in real-world textual-attributed graphs.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. 2023a. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149*.

Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023b. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, pages 417–428.

Donald Loveland, Jiong Zhu, Mark Heimann, Benjamin Fish, Michael T Schaub, and Danai Koutra. 2024. On performance discrepancies across local homophily levels in graph neural networks. In *Learning on Graphs Conference*, pages 6–1. PMLR.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2024a. Reasoning on graphs: Faithful and interpretable large language model reasoning. *Preprint*, arXiv:2310.01061.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. 2024b. Classic gnns are strong baselines: Reassessing gnns for node classification. *arXiv preprint arXiv:2406.08993*.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. 2025. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. *Preprint*, arXiv:2407.10805.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 188–197.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, and 1 others. 2022. Discovering language model behaviors with model-written evaluations. *arXiv preprint arXiv:2212.09251*.

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*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Yide Ran, Zhaozhuo Xu, Yuhang Yao, Zijian Hu, Shanshan Han, Han Jin, Alay Dilipbhai Shah, Jipeng Zhang, Dimitris Stripelis, Tong Zhang, and 1 others. 2024. Alopex: A computational framework for enabling on-device function calls with llms. *arXiv preprint arXiv:2411.05209*.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine

Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, and 1 others. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine*, 29(3):93–93.

Yuan Sui, Yufei He, Nian Liu, Xiaoxin He, Kun Wang, and Bryan Hooi. 2025. Fidelis: Faithful reasoning in large language models for knowledge graph question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8315–8330.

Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2120–2131.

Yuanfu Sun, Zhengnan Ma, Yi Fang, Jing Ma, and Qiaoyu Tan. 2025. Graphicl: Unlocking graph learning potential in llms through structured prompt design. *arXiv preprint arXiv:2501.15755*.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024a. 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.

Jianheng Tang, Qifan Zhang, Yuhan Li, and Jia Li. 2024b. Grapharena: Benchmarking large language models on graph computational problems. *arXiv preprint arXiv:2407.00379*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Gido M Van de Ven and Andreas S Tolias. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. *ICLR (Poster)*, 2(3):4.

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.

Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. 2024b. Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment. *arXiv preprint arXiv:2402.08785*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024c. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Xixi Wu, Yifei Shen, Fangzhou Ge, Caihua Shan, Yizhu Jiao, Xiangguo Sun, and Hong Cheng. 2025. A comprehensive analysis on llm-based node classification algorithms. *arXiv preprint arXiv:2502.00829*.

Lianghao Xia, Ben Kao, and Chao Huang. 2024. Opengraph: Towards open graph foundation models. *arXiv preprint arXiv:2403.01121*.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, and 1 others. 2023. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Z Yang. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, Yongfeng Zhang, and 1 others. 2023. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*, 4(5):7.

Çağatay Yıldız, Nishaanth Kanna Ravichandran, Prishruit Punia, Matthias Bethge, and Beyza Ermis. 2024. Investigating continual pretraining in large language models: Insights and implications. *arXiv preprint arXiv:2402.17400*.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823.

Xingtong Yu, Yuan Fang, Zemin Liu, Yuxia Wu, Zhihao Wen, Jianyuan Bo, Xinming Zhang, and Steven CH Hoi. 2024. A survey of few-shot learning on graphs: from meta-learning to pre-training and prompt learning. *arXiv preprint arXiv:2402.01440*.

Jiawei Zhang. 2023. Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt. *arXiv preprint arXiv:2304.11116*.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

Jiong Zhang, Wei-Cheng Chang, Hsiang-Fu Yu, and Inderjit Dhillon. 2021. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 34:7267–7280.

Siwei Zhang, Yun Xiong, Yateng Tang, Xi Chen, Zian Jia, Zehao Gu, Jiarong Xu, and Jiawei Zhang. 2025a. Unifying text semantics and graph structures for temporal text-attributed graphs with large language models. *arXiv preprint arXiv:2503.14411*.

Yizhuo Zhang, Heng Wang, Shangbin Feng, Zhaoxuan Tan, Xiaochuang Han, Tianxing He, and Yulia Tsvetkov. 2024a. Can llm graph reasoning generalize beyond pattern memorization? *arXiv preprint arXiv:2406.15992*.

Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. 2024b. Llm4dyg: can large language models solve spatial-temporal problems on dynamic graphs? In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4350–4361.

Zhihan Zhang, Xunkai Li, Guang Zeng, Hongchao Qin, Ronghua Li, and Guoren Wang. 2025b. Rethinking graph structure learning in the era of llms. *arXiv preprint arXiv:2503.21223*.

Huanjing Zhao, Beining Yang, Yukuo Cen, Junyu Ren, Chenhui Zhang, Yuxiao Dong, Evgeny Kharlamov, Shu Zhao, and Jie Tang. 2024. Pre-training and prompting for few-shot node classification on text-attributed graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4467–4478.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. 2023. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*.

Huachi Zhou, Jiahe Du, Chuang Zhou, Chang Yang, Yilin Xiao, Yuxuan Xie, and Xiao Huang. 2025. Each graph is a new language: Graph learning with llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17548–17559.

Kerui Zhu, Bo-Wei Huang, Bowen Jin, Yizhu Jiao, Ming Zhong, Kevin Chang, Shou-De Lin, and Jiawei Han. 2024. Investigating instruction tuning large language models on graphs. *arXiv preprint arXiv:2408.05457*.

Chenyi Zi, Haihong Zhao, Xiangguo Sun, Yiqing Lin, Hong Cheng, and Jia Li. 2024. Prog: A graph prompt learning benchmark. *the Thirty-Eighth Advances in Neural Information Processing Systems (NeurIPS 2024)*.

A Comparison between our benchmark and existing works

In Table 9, we summarize the key differences between our benchmarking study and other papers. **Comprehensive Baselines** refers to whether the baseline models cover a wide range of model types. In our paper, we include GNNs, Graph SSL models, Graph Transformers, Foundational Graph Prompt Models, and LLMs with Graph Projectors. **Comprehensive Settings** examines the performance of models across various scenarios, such as vanilla fine-tuning, few-shot learning, and zero-shot learning. **Diverse LLMs** highlights the use of multiple LLMs for comparison, such as Llama, GPT, and Qwen. **LLM Tuning** indicates whether the paper fine-tunes the LLMs or simply uses the original models as they are. Lastly, **Transferability Study** explores whether the paper investigates the cross-task or cross-domain transfer capabilities of the models.

B Related Works

In this section, we review the existing literature on the application of LLMs and related techniques in graph tasks. We highlight two primary categories: the use of LLMs for graph reasoning and their integration with traditional graph models to enhance performance.

B.1 Large Language Models for Graph Reasoning

Recent studies suggest that LLMs have the potential to solve graph reasoning tasks by understanding graph structures (Fatemi et al., 2023; Tang

Model	Node Classification	Link Prediction	Comprehensive Baselines	Comprehensive Settings	Diverse LLMs	LLM Tuning	Transferability Study
InstructGLM (Ye et al., 2023)	✓	✗	✗	✗	✗	✓	✗
LLaGA (Chen et al., 2024b)	✓	✓	✗	✗	✓	✗	✓
InstructGraph (Wang et al., 2024b)	✓	✓	✗	✗	✓	✓	✗
NLGraph (Wang et al., 2024a)	✗	✗	✗	✓	✗	✗	✗
Talk Like a Graph (Fatemi et al., 2023)	✓	✗	✗	✓	✗	✗	✗
All in One (Sun et al., 2023)	✓	✓	✓	✓	✗	✗	✓
OFA (Liu et al., 2023a)	✓	✓	✓	✓	✗	✗	✓
GraphGPT (Tang et al., 2024a)	✓	✓	✗	✗	✓	✓	✓
Ours	✓	✓	✓	✓	✓	✓	✓

Table 9: Comparison between our benchmark and existing works

et al., 2024b). NLGraph (Wang et al., 2024a) indicates that LLMs can track paths within graphs, enabling them to solve tasks such as node connectivity and shortest path detection. Moreover, (Dai et al., 2024a) suggests that LLMs understand graph pattern concepts, which are fundamental to graph structure mining and learning.

Additionally, fine-tuning further enhances the LLMs' reasoning ability in graph tasks (Dai et al., 2024b). (Zhang et al., 2024a) suggest that LLMs can transfer their understanding of substructures through fine-tuning on graphs with different node features. Besides, GraphWiz (Chen et al., 2024a) indicates that LLMs learn path reasoning across various tasks and datasets. Along the same line, FiDeLiS (Sui et al., 2025) leverages stepwise reasoning grounded in knowledge graphs to improve factual consistency in KG-QA, while Think-on-Graph 2.0 (Ma et al., 2025) introduces iterative graph–context retrieval that strengthens LLM reasoning over complex structures. Furthermore, GraphGPT-O (Fang et al., 2025) extends graph reasoning to multimodal attributed graphs, enabling joint image–text generation with structural awareness. These advances highlight that LLMs can be effectively adapted to deepen their comprehension of graph structures.

B.2 Language Model Aided Graph Models

With the advancement of language models, their presence in graph-related tasks has become increasingly prominent. Their natural strengths in language processing and intrinsic reasoning make them particularly valuable, especially in text-attributed graph (TAG) tasks. Broadly, the role of language models in graph learning can be categorized into two main approaches: language models as enhancers and large language models as predictors (Chen et al., 2024c).

B.2.1 Language models as enhancers

Language models serve as enhancers by assisting graph models in representation learning and knowledge integration. Pre-trained language models (PLMs) like BERT (Devlin, 2018), DeBERTa (He et al., 2020), and XLNet (Yang, 2019) are commonly used to transform raw textual descriptions into embeddings, improving graph models' ability to capture node semantics. For instance, OFA (Liu et al., 2023a) encodes text descriptions of nodes and edges into fixed-length vectors, unifying graph data from different domains and enabling strong performance in supervised, few-shot, and zero-shot settings. Similarly, GraphAlign (Hou et al., 2024), BooG (Cheng et al., 2024), and ZeroG (Li et al., 2024a) utilize PLMs to embed textual node features, ensuring feature consistency across diverse datasets during pre-training.

Beyond embedding textual features, large language models (LLMs) contribute to graph representation enrichment. TAPE (He et al., 2023) generates textual explanations for model predictions, which are then transformed into additional node features, enhancing GNN-based learning. On the other hand, LLMGNN (Chen et al., 2023) uses LLMs to annotate a subset of nodes with high quality labels, which are then leveraged by GNNs to predict the remaining unlabeled nodes. This method effectively combines LLMs' semantic reasoning with the structured learning power of GNNs. Building on this direction, Zhou et al. (Zhou et al., 2025) propose to treat each graph as a new language, translating structures into graph–language corpora to enable LLM pre-training that captures structural orders. GraphiT (Khoshraftar et al., 2025) further explores prompt optimization for efficient node classification on TAGs, while Zhang et al. (Zhang et al., 2025b) revisit graph structure learning under the LLM paradigm. For temporal settings, Zhang et al. (Zhang et al., 2025a) unify text semantics with graph structures for temporal TAGs, showing the versatility of LLMs across dynamic scenarios. Likewise, GRAIL (Lim et al., 2025) investigates retrieval-augmented in-context learning, where node embeddings provide graph-aware

contexts to LLMs, improving performance on real-world TAG benchmarks.

B.2.2 Large language models as predictors

LLMs can serve as direct predictors for graph-related tasks such as node classification and link prediction. Instruction tuning is a widely used technique to enhance LLMs' predictive accuracy (Ouyang et al., 2022; Sanh et al., 2021), helping them better interpret graph structures through task-specific prompts. For instance, InstructGLM (Ye et al., 2023) employs multi-prompt tuning to integrate multi-hop structural information, improving its ability to capture complex relationships. GraphGPT (Tang et al., 2024a) follows a dual-stage approach: first, it aligns structural information with language tokens via self-supervised graph matching, and second, it fine-tunes the model on task-specific instructions, leading to more accurate predictions.

Beyond standalone LLMs, hybrid models combine them with GNNs or graph transformers to better leverage graph structure. UniGraph (He and Hooi, 2024) enhances zero-shot learning by aligning textual instructions with category labels while incorporating GNNs for structural learning. GraphLLM (Chai et al., 2023) combines LLM with graph transformer to enrich LLM attention layers with structural and semantic information, enabling more effective graph reasoning. In contrast, LLaGA (Chen et al., 2024b) avoids full LLM tuning and instead fine-tunes a lightweight graph projector, reducing computational cost while maintaining strong predictive performance. These approaches highlight the evolving role of LLMs in graph learning, demonstrating their flexibility in both direct prediction and hybrid architectures.

C Datasets

We summarize the details of used datasets in Table 10. We convert all graphs into undirected graphs and remove self-loops.

For Cora, PubMed, and OGBN-Arxiv, each node represents a paper and the edges denote co-citations. For OGBN-Products, nodes represent Amazon products and edges act as co-purchases. Due to the large size of OGBN-Products, we use Cluster-GCN (Chiang et al., 2019) to process it in smaller partitions. The structural information and label information of these datasets can be achieved from Pyg, and we will release the codes for raw feature processing. Below is some relevant information about each datasets:

- **Cora** (McCallum et al., 2000). Cora has seven categories: ["Rule Learning", "Neural Networks", "Case Based", "Genetic Algorithms", "Theory", "Reinforcement Learning", "Probabilistic Methods"]. The raw text attributes can be obtained from [†].
- **PubMed** (Sen et al., 2008). PubMed has three categories: ["Diabetes Mellitus, Experimental", "'Diabetes Mellitus Type 1", "Diabetes Mellitus Type 2"]. The raw text attributes can be obtained from TAPE (He et al., 2023) [‡].
- **OGBN-Arxiv and OGBN-Products** (Hu et al., 2020). OGB benchmark provides these two datasets. For OGBN-Arxiv, the raw text attributes can be downloaded from [§]. For OGBN-Products, the raw text attributes can be downloaded from [¶].

In extended experiments, we use Computer, Reddit, Instagram, and WikiCS datasets. Computer is from E-Commerce Network, Reddit and Instagram are from Social Networks, and WikiCS represents web link network. We list the details below:

- **Computer**. Computer is from Amazon Electronics dataset (Ni et al., 2019), where each node represents an item in the Computer category. We use the processed dataset released in (Liu et al., 2023a).
- **Reddit and Instagram**. A node represents a user, and edges denote whether two users have replied to each other. The raw text data is collected from (Huang et al., 2024b).
- **WikiCS**. Each node represents a Wikipedia page, and each edge represents a reference link between pages. The raw text data is collected from (Liu et al., 2023a).

Data Split. For node-level tasks, we use the standard train/validation/test splits (Hu et al., 2020): 6:2:2 for Cora, Pubmed, Computer, Reddit, Instagram, and WikiCS, 6:2:3 for the OGBN-Arxiv

[†] https://people.cs.umass.edu/mccallum/data.html

[‡] https://github.com/XiaoxinHe/TAPE

[§] https://snap.stanford.edu/ogb/data/misc/ogbn-arxiv/titleabs.tsv.gz

[¶] http://manikvarma.org/downloads/XC/XMLRepository.html

Dataset	Domain	Task	#Node	#Edge	#Classes	Metrics	Default feature
Cora	citation	Node, Link	2,708	5,429	7	Accuracy	Bag-of-Words (Wang et al., 2024a)
Pubmed	citation	Node, Link	19,717	44,338	3	Accuracy	TF-IDF
OGBN-Arxiv	citation	Node, Link	169,343	1,166,243	40	Accuracy	Skip-gram (Mikolov et al., 2013)
OGBN-Products	e-commerce	Node, Link	2,449,029	61,859,140	47	Accuracy	Bag-of-Words
Computer	e-commerce	Node	87,229	721,081	10	Accuracy	-
Reddit	social network	Node	33,434	198,448	2	Accuracy	-
Instagram	social network	Node	11,339	144,010	2	Accuracy	-
WikiCS	web link	Node	11,701	215,863	10	Accuracy	-

Table 10: Datasets

dataset ,and 8:2:90 for OGBN-Products. For link prediction, we randomly sample node pairs from the training nodes for training and from the testing nodes for evaluation. The size of the edge-level training set matches that of the node-level training set.

D Impacts of Different Node Feature Embedding Methods

Node features play a crucial role in node classification and link prediction tasks. For LLMs, raw text attributes are directly used as node features, while datasets like Cora, PubMed, Arxiv, and Products provide default preprocessed features generated through feature embedding methods (as shown in Table 10). This raises an important question: **is it fair to compare baseline models using default features with LLMs that rely on raw text attributes?**

To address this, we embedded the raw text attributes using various pre-trained LLMs and fed these embeddings into GraphSAGE for node classification tasks. The results are summarized in Table 11. Specifically, all-MiniLM-L6-v2 is the latest Sentence-BERT model, and text-embedding-ada-002 is the latest embedding model from OpenAI.

From the results, we observe no significant accuracy improvements when using pre-trained LLM embeddings over the default node features. In some datasets, LLM-based embeddings perform better, while in others, default node features yield stronger results. Therefore, we believe that using the default node features provided by corresponding datasets is reasonable and fair.

E Detailed Experimental Settings

E.1 Computation Environment

In this paper, all the experiments were conducted on one single server with 4 80G Nvidia A100 GPUs.

Embedding Methods	Cora	PubMed	Arxiv	Products
default	89.67	89.02	71.35	82.89
all-MiniLM-L6-v2 (Reimers, 2019)	89.88	89.91	72.03	81.82
t5-small (Raffel et al., 2020)	86.71	87.78	70.28	79.64
e5-base (Wang et al., 2022)	88.10	87.12	71.52	80.33
text-embedding-ada-002	89.30	89.72	72.20	82.45

Table 11: Impacts of different node feature embedding methods. Task: node classification. Model: GraphSAGE

E.2 Model Settings

- **GCN & GraphSAGE**

```
num_layers=3, hidden_channels=256,
    dropout=0.5,
norm='batchnorm', activation='relu',
optimizer=torch.optim.AdamW, lr=0.005,
     weight_decay=1e-4,
scheduler=torch.optim.lr_scheduler.
    StepLR, step_size=20, gamma=0.5,
patience=20, min_delta=1e-3, epochs
    =8000
```

- **GAT**

```
num_layers=3, hidden_channels=256,
    dropout=0.5, heads=2,
norm='batchnorm', activation='relu',
optimizer=torch.optim.AdamW, lr=0.005,
     weight_decay=1e-4,
scheduler=torch.optim.lr_scheduler.
    StepLR, step_size=20, gamma=0.5,
patience=20, min_delta=1e-3, epochs
    =8000
```

- **MixHop**

```
num_layers=2, hidden_channels=256,
    powers=[ [0,1,2], [0,1] ], dropout
    =0.6,
add_self_loops=True, activation='relu
    ', aggregation='mixhop',
```

```
optimizer=torch.optim.AdamW, lr=0.005,
     weight_decay=1e-4,
scheduler=torch.optim.lr_scheduler.
    StepLR, step_size=20, gamma=0.5,
early_stopping=dict(patience=20,
    min_delta=1e-3), max_epochs=8000,
    log_interval=10
```

- **GraphCL**

```
Graph Encoder:
    -Backbone: GCN, -Hidden Channels:
        128, -Activation: ReLU, -
        Optimizer: Adam, -lr=0.01, -
        Epochs: 100
Data Augmentations:
    -Feature Masking: mask_rate=0.3, -
        Edge Perturbation: perturb_rate
        =0.1
Contrastive Loss:
    -Normalization: L2 (dim=1), -
        Temperature: 0.5
Linear Classifier:
    -Input Features: 128, -Optimizer:
        Adam, -lr=0.01, -Epochs: 50 (
        supervised training)
```

- **GraphMAE**

```
Graph Encoder:
    -Backbone: GCN, -Hidden Channels:
        256, -Activation: ReLU, -
        Optimizer: Adam, -lr=0.01, -
        Epochs: 200
Data Augmentations:
    -Feature Masking: mask_ratio=0.5 (
        encoder-level), -Random Masking
        : mask_rate=0.3 (training-level
        )
Reconstruction Loss:
    -Loss Function: MSE Loss, -
        Reconstruction Target: Masked
        node features
Linear Classifier:
    -Input Features: 256, -Optimizer:
        Adam, -lr=0.01, -Epochs: 100 (
        supervised training)
```

- **Graphormer** We follow the hyper-parameter settings in the original paper (Ying et al., 2021).

- **Prodigy** We follow the hyper-parameter settings in the original paper (Huang et al., 2024a).

- **OFA** We follow the hyper-parameter settings in the original paper (Liu et al., 2023a).

- **GIANT & TAPE**

```
gnn_type='GraphSAGE', num_layers= [2,
    3, 4], hidden_channels= [128, 256],

optimizer=torch.optim.Adam, lr=0.001,
    weight_decay=0, dropout= [0.3, 0.5,
     0.6]
```

- **All in one & GPF-plus & GraphPrompt**

```
gnn_type='GCN', num_layers=2,
    hidden_channels=128, JK='last',
prompt_type=['All in one', 'GPF-plus',
     'GraphPrompt'],
optimizer=torch.optim.Adam, lr=0.001,
    weight_decay=0, dropout=0.5,
epochs=800, batch_size=128, shot_num
    =5,
```

For detailed prompt designs, we follow the original papers (Sun et al., 2023), (Fang et al., 2024), and (Liu et al., 2023b).

- **ZeroG** We follow the hyper-parameter settings in the original paper (Li et al., 2024a).

- **LLaGA** We follow the hyper-parameter settings in the original paper (Chen et al., 2024b).

- **Llama3B & Llama8B**

```
LLM Configuration:
    -Base Model: [meta-llama/Llama
        -3.2-3B-Instruct, meta-llama/
        Llama-3.1-8B-Instruct],
    -Use LoRA: true, -Max Sequence
        Length: 1024, -Model Precision:
         bfloat16
LoRA Configuration:
    -LoRA Rank (r): 16, -LoRA Alpha:
        32, -LoRA Dropout: 0.05,
    -Target Modules: [o_proj,
        gate_proj, down_proj, up_proj]
Training Configuration:
    -Optimizer: adamw_torch, -Learning
         Rate: 4e-4, -Train Batch Size:
         2 x 12 (per_device x
        grad_accum),
    -Total Epochs: 1, -Gradient Accu
        Steps: 12, -Pad Token ID: -100
        (IGNORE_INDEX)
```

```
DeepSpeed Optimization:
    -Zero Stage: 2, -Offload Strategy:
        [-Optimizer -> CPU (pinned) ,-
        Activation Checkpointing: true
        ],
    -Pipeline Parallel: [-Enabled:
        true, -Micro Batch Size: 1]
Data Processing:
    -Data Sources: [Cora, PubMed,
        Arxiv, Products], -Input Format
        : System Prompt + User Query +
        Answer,
    -Data Limits: [-Product/node: max
        3,000 samples, -Product/link:
        max 2,000 samples],
    -Preprocessing Workers: 20,
    -Cora & PubMed & Arxiv: [-Max 1-
        hop neighbors: 20, -Max 2-hop
        neighbors: 5],
    -Products: [-Max 1-hop neighbors:
        30, -Max 2-hop neighbors: 10]
```

F Hyperparameter Search Space

For transparency and reproducibility, this section details the hyperparameter search spaces explored during the tuning of our baseline models. The final parameters selected for the experiments are reported in Appendix E. All selections were made based on the best-performing validation accuracy.

F.1 GNN Models (GCN, GraphSAGE, GAT)

For the graph neural network baselines, we performed a grid search over the following hyperparameters on each dataset:

- **Learning Rate**: {0.01, 0.005, 0.001}
- **Hidden Channels**: {128, 256}
- **Number of Layers**: {2, 3}
- **Dropout Rate**: {0.3, 0.5, 0.6}
- **Weight Decay**: {5e-4, 1e-4, 5e-5}

F.2 LoRA Fine-Tuning for Llama Models

For the instruction tuning of Llama3B and Llama8B using Low-Rank Adaptation (LoRA), we explored a focused set of hyperparameters known to be effective for this technique. The primary goal was to find a stable configuration without conducting an exhaustive, computationally prohibitive search.

- **Learning Rate**: {4e-4, 2e-4, 1e-4}
- **LoRA Rank (r)**: {8, 16, 32}
- **LoRA Alpha** (α): {16, 32}. We maintained the common practice of setting alpha to be twice the rank.
- **LoRA Dropout**: {0.05, 0.1}

The target modules for LoRA (o_proj, gate_proj, etc.) were kept consistent across all runs as they are standard choices for Llama-family models. The final chosen configuration was a learning rate of 4e-4, a rank of 16, an alpha of 32, and dropout of 0.05, which provided a good balance of performance and stability.

G Baseline models

In this paper, we evaluate multiple baseline models and provide detailed descriptions of their implementations as follows. These models were applied to a consistently preprocessed version of the datasets to ensure fair comparisons and produce the experimental results presented in this study.

1. **GNNs**: For GCN (Kipf and Welling, 2016), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2017), and Mixhop (Abu-El-Haija et al., 2019), we follow the models on OGB Leaderboards [‖]. Specifically, the first three models are all from (Luo et al., 2024b), and the codes can be obtained from [**].

2. **Graph SSL Models**: We choose GraphCL (You et al., 2020) and GraphMAE (Hou et al., 2022) in this categories. GraphCL employs contrastive learning by distinguishing augmented views of the same graph from others, while GraphMAE uses masked autoencoding, reconstructing masked graph components to learn node representations without requiring augmented views. For GraphCL, we follow the implementation from [††]. For GraphMAE, we follow the implementation from [‡‡].

3. **Graph Transformers**: We use Graphormer (Ying et al., 2021) in this categories. Graphormer is a transformer-based model designed specifically to handle graph-structured

[‖] https://ogb.stanford.edu/docs/leader_nodeprop/
[**] https://github.com/LUOyk1999/tunedGNN
[††] https://github.com/Shen-Lab/GraphCL
[‡‡] https://github.com/THUDM/GraphMAE

data, enabling efficient processing and analysis of complex relational information.The implementation is from .

4. **Foundational Graph Prompt Models**: We use Prodigy (Huang et al., 2024a), OFA (Liu et al., 2023a), All in one (Sun et al., 2023), GPF-plus (Fang et al., 2024), GraphPrompt (Liu et al., 2023b), and ZeroG (Li et al., 2024a) in this categories.
   - Prodigy enables in-context learning over graphs by utilizing a novel prompt graph representation and a family of in-context pretraining objectives, achieving superior performance on diverse downstream classification tasks without the need for retraining.
   - OFA represents nodes and edges as human-readable text, mapping them from various domains into a unified space using LLMs. The framework then adapts to different tasks by embedding task-specific prompts within the input graph.
   - All in one proposes a novel method to unify graph prompts and language prompts, enhancing the performance of various graph tasks through effective prompt design and meta-learning techniques.
   - GPF-plus is an enhanced graph prompt tuning method that assigns independent learnable vectors to each node, offering great flexibility and expressiveness and consistently outperforming other methods in various experiments.
   - GraphPrompt leverages a common task template based on subgraph similarity, enhanced with task-specific learnable prompts to improve performance across different tasks such as node and graph classification.
   - ZeroG uses a language model to encode node features and class labels, incorporating prompt-based subgraph sampling and efficient fine-tuning techniques to tackle the challenges of cross-dataset zero-shot transferability in graph learning.

   The implementations of Prodigy and OFA can be obtained from and , respectively. For All in one, GPF-plus, and GraphPrompt, we use the implementation from ProG (Zi et al., 2024) . For ZeroG, we follow the implementation from .

5. **LM-Augmented Graph Learning Models**: We choose GIANT (Chien et al., 2021) and TAPE (He et al., 2023). GIANT conducts neighborhood prediction using XR-Transformers (Zhang et al., 2021), resulting in an LLM that generates superior feature vectors for node classification compared to traditional bag-of-words and standard BERT embeddings. TAPE uses explanations from LLMs as features to enhance the performance of GNNs on text-attributed graphs, achieving state-of-the-art results on various benchmarks with significantly lower computation time. For GIANT and TAPE, we follow the implementation from .

6. **LLM with Graph Projectors**: LLaGA (Chen et al., 2024b) is chosen for this category. The implementation is from .

H Theoretical Justification: How LLMs Emulate Graph Reasoning

A primary concern regarding the application of LLMs to graph tasks is the fundamental modality mismatch: LLMs are pre-trained on sequential text, whereas graphs are inherently non-sequential, relational data structures. This section provides a theoretical and mechanistic justification for why LLMs, particularly through the mechanisms of prompting and instruction tuning, can successfully perform graph reasoning. Our core argument is that **the Transformer's self-attention mechanism can be viewed as a powerful and general form of a graph operator, which learns to emulate the message-passing operations of Graph Neural Networks (GNNs) when guided by structured textual prompts and task-specific fine-tuning.**

H.1 The Self-Attention Mechanism as a General Graph Operator

The Transformer architecture is fundamentally a relational reasoner. For a sequence of n token embeddings $X \in \mathbb{R}^{n \times d}$, the self-attention mechanism computes a new set of representations $Z \in \mathbb{R}^{n \times d}$

https://github.com/microsoft/Graphormer
https://github.com/snap-stanford/prodigy
https://github.com/LechengKong/OneForAll
https://github.com/sheldonresearch/ProG

https://github.com/NineAbyss/ZeroG
https://github.com/NineAbyss/GLBench
https://github.com/VITA-Group/LLaGA

as:

$$Z = \text{Attention}(Q, K, V) = \underbrace{\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)}_{A} V \tag{1}$$

The attention matrix $A \in \mathbb{R}^{n \times n}$ contains pairwise scores A_{ij} representing the influence of token j on token i. This allows us to interpret the attention mechanism as performing an update on a **dynamic, fully-connected, weighted graph** $\mathcal{G}_A = (\mathcal{T}, A)$, where the set of tokens $\mathcal{T}$ are the nodes.

The representation z_i for the i-th token is an aggregation over all tokens in the sequence:

$$z_i = \sum_{j=1}^{n} A_{ij} v_j \tag{2}$$

This is a generalized form of neighborhood aggregation, where the "neighborhood" of each token is the entire sequence, and the weights are learned based on context.

H.2 Emulating GNN Message Passing via Prompt-Structured Attention

The success of GNNs stems from the message-passing paradigm. For a node v in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the update rule for its hidden representation $h_v^{(l)}$ at layer l is:

$$m_{\mathcal{N}(v)}^{(l+1)} = \text{AGGREGATE}^{(l)}\left(\{h_u^{(l)} : u \in \mathcal{N}(v)\}\right) \tag{3}$$

$$h_v^{(l+1)} = \text{UPDATE}^{(l)}\left(h_v^{(l)}, m_{\mathcal{N}(v)}^{(l+1)}\right) \tag{4}$$

We posit that an LLM, when given a structured textual prompt, emulates this two-step process. Let the input prompt serialize a target node v and its neighborhood $\mathcal{N}(v)$. The set of all tokens $\mathcal{T}$ can be partitioned into three disjoint sets: tokens representing the target node itself ($\mathcal{T}_v$), tokens representing its neighbors ($\mathcal{T}_{\mathcal{N}(v)}$), and all other tokens ($\mathcal{T}_{\text{other}}$).

The attention-based update for a token $i \in \mathcal{T}_v$ can then be decomposed as:

$$z_i = \underbrace{\sum_{j \in \mathcal{T}_v} A_{ij} v_j}_{\text{Self-update}} + \underbrace{\sum_{j \in \mathcal{T}_{\mathcal{N}(v)}} A_{ij} v_j}_{\text{Neighbor Aggregation}} + \underbrace{\sum_{j \in \mathcal{T}_{\text{other}}} A_{ij} v_j}_{\text{Contextual Noise/Signal}} \tag{5}$$

This decomposition reveals the analogy to GNNs:

- **Aggregation:** The term $\sum_{j \in \mathcal{T}_{\mathcal{N}(v)}} A_{ij} v_j$ is a direct analogue to the GNN **AGGREGATE** function (Eq. 3). The LLM learns to assign high attention scores A_{ij} to tokens representing the true neighbors of node v, effectively aggregating their information.
- **Update:** The aggregated message is then combined with the node's own representation (the self-update term) and passed through a position-wise Feed-Forward Network (FFN):

  $$h_i' = \text{FFN}(\text{LayerNorm}(z_i + x_i)) \tag{6}$$

  This FFN, a powerful non-linear transformer, serves as the **UPDATE** function (Eq.4), producing the final, contextually-aware representation for the token.

By providing k-hop neighbor information, we allow the LLM to implicitly simulate a k-layer GNN within a single forward pass.

H.3 Learning Graph Inductive Biases via Instruction Tuning

An off-the-shelf LLM, while architecturally capable, lacks the specific *inductive biases* for graph structures. It has no inherent reason to prioritize tokens labeled "neighbor" over any other tokens. **Instruction tuning** instills these biases by optimizing the model's parameters θ on a graph-specific objective.

Let $\mathcal{D}_{\text{graph}} = \{(\text{prompt}(G_k, v_k), y_k)\}_{k=1}^{M}$ be a dataset of graph-based instruction-response pairs. The tuning process minimizes a loss function, typically the cross-entropy for classification tasks:

$$\mathcal{L}_{\text{tune}}(\theta) = -\sum_{k=1}^{M} \log P(y_k | \text{prompt}(G_k, v_k); \theta) \tag{7}$$

The key effect of minimizing $\mathcal{L}_{\text{tune}}$ is the reshaping of the attention matrix A. The optimization process implicitly forces the attention patterns to reflect the graph structure. Specifically, for a target node token i and a neighbor token j, the fine-tuning process encourages:

$$\theta^* = \arg\min_\theta \mathcal{L}_{\text{tune}}(\theta) \implies A_{ij} \text{ is high if token } j \in \mathcal{T}_{\mathcal{N}(v_i)} \tag{8}$$

Essentially, instruction tuning teaches the LLM that for graph-related prompts, the "correct" reasoning path involves focusing attention on the explicitly provided neighborhood information. It transforms the generic, semantically-driven attention into a specialized, structurally-aware attention mechanism that mimics the sparse connectivity of the original graph.

H.4 Synthesis: When and Why LLMs Succeed or Falter

This theoretical framework helps explain our empirical findings:

- **Success in Local Tasks:** LLMs excel at tasks like node classification and link prediction because these primarily rely on local structures (1-hop, 2-hop), which can be effectively encoded and processed by the partitioned attention mechanism described above.
- **Impact of Instruction Tuning:** The significant performance gap between pre-trained and tuned LLMs is explained by the optimization of $\mathcal{L}_{\text{tune}}$, which is necessary to instill graph-centric inductive biases into the model's attention patterns.
- **Potential Limitations:** This framework also predicts limitations. Tasks requiring **global graph properties** (e.g., diameter, global clustering coefficient) are challenging because the full structure, and thus the complete attention graph, cannot fit into a finite context window. Furthermore, for purely algorithmic tasks (e.g., shortest path), LLMs act as probabilistic pattern matchers rather than deterministic solvers, leading to approximations. Classical algorithms, which operate on the true graph adjacency matrix, remain more precise and efficient for such problems.

In conclusion, the success of LLMs in graph tasks is a direct consequence of the Transformer architecture's inherent ability to model relational data. This latent ability is unlocked and specialized through instruction tuning, which aligns the self-attention mechanism to emulate the message-passing framework of GNNs.

I In-depth Analysis of LLM Sensitivity to Prompt Variations

I.1 Motivation

A critical observation from our main experiments is that the performance of LLMs on graph learning tasks is highly sensitive to the prompt format. To address the reviewer's feedback for a deeper understanding of this phenomenon, this section presents a systematic investigation into *why* this sensitivity exists. We aim to dissect the components of a prompt and analyze their individual impact on the model's reasoning process for node classification.

I.2 Experimental Design

To isolate the factors influencing performance, we conducted a controlled experiment with the following setup:

- **Model:** We use **Llama8B** as our representative model.
- **Task and Dataset:** We focus on the **node classification** task on the **Cora** dataset, allowing for a focused and granular analysis.
- **Methodology:** We establish a **baseline prompt** using the exact performance reported in our main experiments (Table 1). We then introduce a series of controlled, single-factor variations. By modifying only one aspect of the prompt at a time, we attribute any resulting performance change directly to that modification.

We designed three categories of prompt variations to test distinct hypotheses about LLM sensitivity:

1. **Instruction Verbosity:** How does the level of detail in the task description affect the model's focus and understanding?
2. **Structural Phrasing:** How crucial are explicit graph-related terms (e.g., "neighbor") compared to more natural, relational phrasing?
3. **Task Command Wording:** How robust is the model to semantic but trivial paraphrasing of the core instruction?

I.3 Results and Analysis

The performance of Llama-8B under these controlled prompt variations is summarized in Table 12. Our analysis of these results reveals several key insights into why prompt format is so influential:

1. Clarity and Sufficient Context are Crucial. The *Minimal Instruction* prompt, which strips away the guiding context (e.g., "You are a good graph reasoner..."), causes a performance drop of 3 percentage points. This suggests that LLMs benefit from "role-playing" or context-setting instructions that frame the problem. Without this frame, the model may struggle to activate the most relevant reasoning pathways. Conversely, the *Verbose Instruction* prompt, while providing more detail, slightly underperforms the baseline. This indicates a trade-off: while some context is essential, excessive detail may introduce irrelevant information that dilutes

Variation Category	Prompt Description	Accuracy (%)
Baseline	**Standard "1-hop w/o label" prompt**	**58.4**
Instruction Verbosity	*Minimal Instruction*: "Given the target and neighbors, predict its category."	55.5
	Verbose Instruction: "You are an expert paper classifier..."	56.8
Structural Phrasing	*Relational*: "...target paper is connected to the following papers..."	55.1
	Implicit: "Target Paper: ... Related Papers: ..."	53.7
Task Command Wording	*Synonym 1*: "...assign the correct label to the Target node."	58.9
	Synonym 2: "What is the research area of the Target node?"	58.1

Table 12: Performance analysis of Llama-8B on the Cora node classification task under different prompt variations. The baseline is the standard "1-hop w/o label" prompt.

the focus on the core data, acting as noise. The standard baseline prompt appears to strike an effective balance.

2. Explicit Structural Language Bridges the Gap between Text and Graphs. The most significant finding comes from the *Structural Phrasing* variations. Replacing the explicit, technical term "Known neighbor papers at hop 1" with more natural language like "connected to" (*Relational*) or simply "Related Papers" (*Implicit*) consistently degrades performance. The accuracy drop is most severe with the implicit phrasing. This strongly suggests that LLMs, being pre-trained on sequential text, do not inherently interpret a list of items following a target as a formal graph structure. Explicit keywords like **"hop"** and **"neighbor"** act as powerful signals that help the model shift from a standard text-processing mode to a graph-reasoning mode. These terms provide an unambiguous structural scaffold that is otherwise missing, forcing the model to recognize and leverage the relational nature of the input data.

3. LLMs are Robust to Simple Paraphrasing of the Core Task. The *Task Command Wording* variations show almost no change in performance. Replacing "predict the category" with "assign the label" or phrasing it as a question ("What is the research area...") results in nearly identical accuracy. This demonstrates that the model has a robust semantic understanding of the core objective. Its sensitivity is not rooted in surface-level vocabulary for the main task command, but rather in the description and framing of the input data's structure.

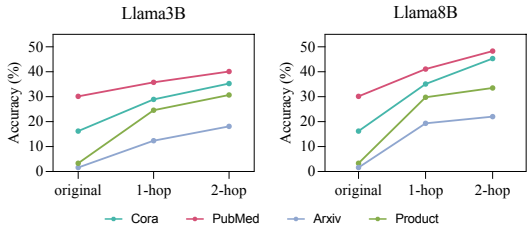

Figure 3: LLM performance on node classification without node attributes

J Extended Experiments

J.1 LLM Understanding of Graph Structures

The structure of a graph sets it apart from natural language, and the model ability to comprehend these structures is vital for enhancing its performance on graph tasks. In this section, we explore the ability of instruction-tuned LLMs to understand graph structures.

J.1.1 Experiment Settings

We remove all node attributes and retain only node IDs to eliminate the influence of attributes on LLM reasoning. Examples of these prompt formats are provided in Appendix L.3.

J.1.2 Results

Node classification We present the results of node classification in Figure 3. "Original" refers to Llama3B or Llama8B without parameter optimization, while "1-hop" and "2-hop" correspond to 1-hop w/o label and 2-hop w/o label, respectively. From the figure, we observe that off-the-shelf LLMs perform similarly to random guessing in node classification. For instance, with 7 classes in Cora, the probability of random guessing correctly is 14.28%, and the experimental results align closely with this probability. This is because LLMs struggle to make accurate predictions based

purely on graph structure without semantic information. After instruction tuning, LLMs start to learn some graph structural information, leading to improved accuracy. However, the improvement is limited, likely because the classes in these datasets are strongly correlated with node features, and the graph structural differences between categories are minimal. This explains why simpler models like MLPs (Hu et al., 2021b) and our ego prompt format perform relatively well, as they rely more on the node features than on the graph structure itself.

Models	Prompts	Cora	PubMed	ArXiv	Products	Avg
Llama3B w attributes	1-hop	72.97	71.55	72.45	78.92	73.97
	2-hop	68.21	59.95	68.55	79.17	68.97
Llama8B w attributes	1-hop	80.44	74.80	87.80	85.29	82.08
	2-hop	89.39	77.30	92.30	90.77	87.44
Llama3B w/o attributes	1-hop	66.61	55.44	64.94	78.47	66.37
	2-hop	72.22	58.62	65.62	74.52	67.75
Llama8B w/o attributes	1-hop	63.19	55.81	68.62	81.30	67.23
	2-hop	85.58	69.50	84.88	87.78	81.94
tuned Llama3B w/o attributes	1-hop	75.88	74.70	78.30	77.38	76.57
	2-hop	93.20	**97.66**	89.00	94.09	93.49
tuned Llama8B w/o attributes	1-hop	85.15	78.81	89.34	87.98	85.32
	2-hop	**94.11**	97.44	**93.67**	**94.54**	**94.94**

Table 13: LLM performance on link prediction without node attributes. Llama3B w attributes and Llama8B w attributes are for comparison. The **best** and second-best are highlighted.

Link prediction From Table 13, we observe that LLMs with node attributes outperform those without, highlighting the positive role of node attributes in LLM reasoning. However, after instruction tuning without node attributes, the LLMs show a significant improvement in link prediction accuracy. This demonstrates that LLMs can effectively learn and understand graph structures, achieving high link prediction accuracy even in the absence of node attributes.

Remark 5. *LLMs can learn graph structures effectively through instruction tuning. While node attributes improve performance, LLMs can still achieve high accuracy in link prediction by leveraging structural information alone. However, the improvement in node classification is limited, likely because the classes are closely related to node features and the structural differences between categories are minimal.*

J.2 Robustness of LLMs

We aim to investigate the robustness of LLMs under two challenging conditions: missing edge information and decreasing graph homophily. Graph homophily refers to the tendency of similar nodes to connect. Our goal is to understand whether LLMs primarily rely on node similarity when performing graph reasoning and how reducing this similarity affects their performance.

J.2.1 Experiment Settings

We conduct experiments on the **Cora** and **ArXiv** datasets, designing two scenarios: **drop same** and **drop random**. The former examines how reducing node similarity affects LLM performance, while the latter investigates the impact of simply reducing the number of edges.

- **Drop Same**: We randomly remove **0%, 20%, 40%, 60%, 80%, and 100%** of edges connecting nodes of the same class. This reduces node similarity, effectively lowering the homophily ratio (Loveland et al., 2024; Huang et al., 2023).
- **Drop Random**: We randomly remove edges but have to ensure that the number of dropped edges matches the corresponding **"drop same"** setting. For example, if 40% "drop same" results in 1,000 removed edges, then 40% "drop random" also removes 1,000 edges.

For LLMs, we use **DeepSeek V3** and **Llama3B**. As baselines, we include **GCN**, **GraphSAGE**, and **MixHop** (Abu-El-Haija et al., 2019) (which performs well on heterophilic graphs). All trainable models (GCN, GraphSAGE, MixHop, and Llama3B) are trained on graphs with varying levels of edge removal. Specifically, for each dataset (Cora and ArXiv), we train **twelve** models per method—six under "drop same" and six under "drop random", corresponding to the six drop percentages.

J.2.2 Results

We summarize the experimental results in Figure 4. As expected, accuracy declines across all models and datasets as the edge drop percentage increases. However, the impact of edge removal is not uniform. The "drop same" condition leads to a sharper decline compared to "drop random", suggesting that reducing node similarity (homophily) has a greater negative effect than simply removing edges at random.

Interestingly, DeepSeek V3 and tuned Llama3B show more resilience to homophily reduction compared to GCN, GraphCL, and even Mixhop, indicating that they rely less on node similarity for

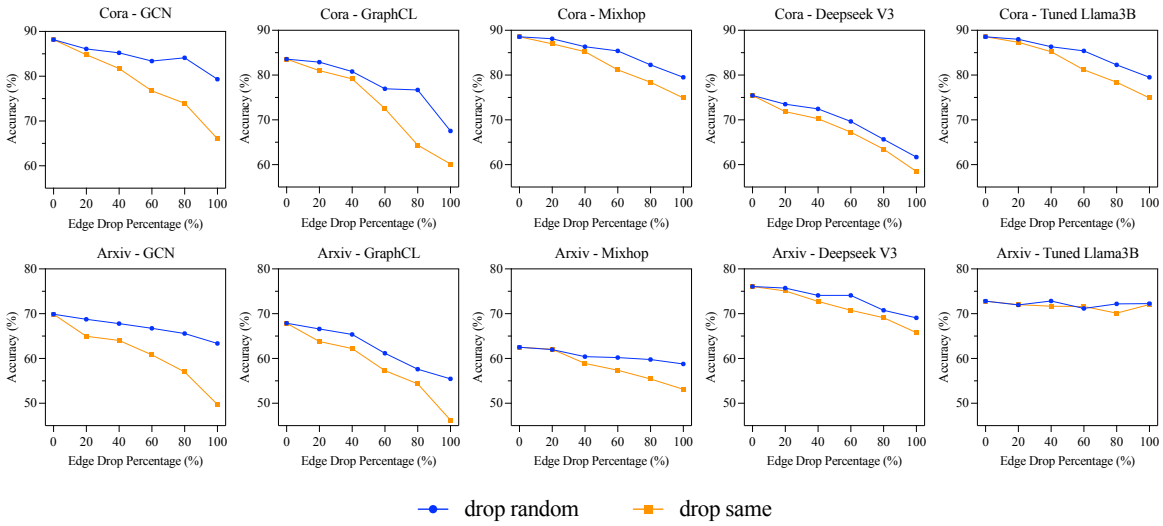

Figure 4: Robustness of LLMs

classification. Among them, tuned Llama3B stands out, not only preserving high accuracy despite edge removal but also showing the lowest dependency on node similarity. This highlights that instruction tuning significantly enhances the robustness of LLMs, making them more adaptable to structural perturbations.

Remark 6. *Reducing homophily (via "drop same") has a more significant negative impact than randomly removing edges. LLMs, especially those after instruction tuning are more resilient to structural perturbations compared to GNNs like GCN, GraphSAGE, and even MixHop.*

J.3 Computational Overhead Analysis

Computational overhead is an important consideration for real-world deployment. We evaluate both the training and inference times of several baseline models and LLMs with instruction tuning. The results are presented in Table 14 and Table 15. All measurements were conducted on a single NVIDIA A100-80G GPU.

Based on the results, we observe that during training, graph-specific models incur significantly lower computational overhead compared to LLM-based methods. For instance, the training time of Llama8B exceeds that of classic GNNs by more than 100×. This highlights the importance of LLM transferability (discussed in Section 5.3: if a one-time training process can support multiple downstream tasks, the high training cost may be justified. Therefore, a promising research direction is to further improve the adaptability and generalization of LLMs across different graph domains and tasks.

As for inference, although LLM-based models still require more time than graph-specific ones, the difference is less critical since all inference times are within the millisecond range. Thus, unless strict real-time performance is required, the overhead gap is relatively negligible.

J.4 Comparison of Different LLMs on Node Classification

In Section 3, we provided a detailed summary of the performance of Llama3B, Llama8B, and Qwen-plus on the node classification task. This served as a foundation for understanding how different model sizes and architectures influence performance on graph-related problems. In this subsection, we expand our exploration by introducing additional large language models (LLMs) and examining diverse prompt formats.

J.4.1 Experiment Settings

We compare the performance of **Llama3B** (Touvron et al., 2023), **Llama8B** (Touvron et al., 2023), **Qwen3-32B** (Yang et al., 2025), **Qwen-plus** (Bai et al., 2023), **Qwen-max** (Bai et al., 2023), **GPT-4o** (Achiam et al., 2023), **Deepseek V3** (Liu et al., 2024), and **Gemini2.5 Pro** (Comanici et al., 2025) on node classification tasks in the **ego scenario**, where no structural information about the target node is provided. The evaluation uses four distinct prompt formats: the original prompt, Chain of Thought (CoT) (McCallum et al., 2000), Build A Graph (BAG) (Wang et al., 2024a), and in-context few-shot. Below, we provide a brief overview of

Models	Node classification				Link prediction			
	Cora	PubMed	ArXiv	Products	Cora	PubMed	ArXiv	Products
GCN	15.9s	34.9s	8.4m	15m	9.5s	26.8s	7.7m	13.4m
GraphSAGE	14.1s	33.3s	7.8m	13.8m	8.2s	23.7s	7.2m	12m
GAT	20.5s	45s	10.1m	18.4m	10.8s	32.9s	8.1m	15.2m
GraphCL	2.1m	4.1m	53m	1.2h	2m	3.2m	48.2m	1.1h
GraphMAE	3.8m	6.3m	1.1h	1.5h	3.2m	4.9m	1h	1.4h
LLaGA	13.8m	29m	6h	8.4h	12m	27.5m	5.2h	7.9h
Llama3B	1.2h	1.9h	18.3h	23.9h	1.7h	2.3h	23.2h	26.9h
Llama8B	1.8h	2.6h	25.7h	31.1h	2.1h	3h	30h	35.8h

Table 14: Training times of different models on node classification and link prediction tasks. We use 9 prompt formats to train LLMs on link prediction. LLM tuning was done on 4 A100-80G GPUs, so all reported times are multiplied by 4.

Models	Node classification				Link prediction			
	Cora	PubMed	ArXiv	Products	Cora	PubMed	ArXiv	Products
GCN	8ms	12ms	33ms	40ms	3ms	5ms	26ms	38ms
GraphSAGE	12ms	29ms	35ms	3ms	4ms	24ms	27ms	33ms
GAT	8ms	10ms	35ms	38ms	4ms	5ms	29ms	41ms
GraphCL	12ms	19ms	69ms	71ms	7ms	10ms	50ms	71ms
GraphMAE	15ms	22ms	76ms	80ms	8ms	11ms	57ms	69ms
LLaGA	40ms	69ms	112ms	159ms	27ms	37ms	99ms	134ms
Llama3B	186ms	211ms	338ms	401ms	113ms	139ms	172ms	228ms
Llama8B	231ms	238ms	381ms	459ms	127ms	151ms	203ms	273ms

Table 15: Inference times of different models on node classification and link prediction tasks.

each prompt format:

- **Original Prompt:** This prompt is identical to the one used in Section 3. It provides the basic context and query format for node classification tasks. Specific examples can be found in Table 17.

- **CoT:** Based on the original prompt, this format appends the instruction *"Let's think step by step"* to encourage the model to output a structured reasoning process in a step-by-step manner.

- **BAG:** Building upon the original prompt, this format adds the instruction *"Let's construct a graph with the nodes and edges first"*. This is designed to guide the model toward constructing an implicit graph representation before reasoning about the classification task.

- **In-Context Few-Shot:** This format supplements the original prompt with three concrete question-answer examples. These examples aim to provide additional context and demonstrate how similar tasks should be handled.

J.4.2 Results

We summarize the results in Table 16. The overall trend suggests that larger models tend to perform better. For instance, Llama8B consistently outperforms Llama3B, and Qwen-max generally achieves higher accuracy than Qwen-plus.

Across most models, CoT improves performance over the original prompt in Cora and PubMed, particularly for smaller models like Llama3B and Llama8B. This suggests that breaking down the reasoning process helps the model make better predictions. However, on ArXiv and Products, CoT leads to performance degradation. One possible reason is that small-class datasets (like Cora and PubMed) have clear category boundaries, making structured reasoning effective. In contrast, large-class datasets (like ArXiv and Products) have high inter-class similarity, increasing ambiguity. In such cases, CoT may introduce erroneous reasoning steps by misassociating nodes with semantically similar classes.

BAG results in significant accuracy drops for smaller models (e.g. Llama3B and Llama8B), while larger models show more stability but still do not outperform CoT or in-context few-shot. This could be due to the additional reasoning complexity

Model	Prompt	Cora	PubMed	ArXiv	Products	Avg
Llama3B	original	24.72	63.20	23.10	40.80	37.96
	CoT	**42.19**	**71.43**	**29.90**	**50.21**	**48.43**
	BAG	15.68	35.32	2.00	30.00	20.67
	in-context few-shot	39.48	62.20	25.63	42.85	42.52
Llama8B	original	43.39	77.80	**59.35**	50.12	57.67
	CoT	**53.51**	**81.80**	53.24	47.41	58.99
	BAG	23.80	21.08	5.80	32.13	20.68
	in-context few-shot	51.29	80.13	54.60	**52.20**	**59.41**
Qwen3-32B	original	48.33	80.20	66.41	61.30	64.06
	CoT	**57.92**	81.69	**68.10**	**63.70**	**67.85**
	BAG	50.30	84.90	64.96	60.31	65.12
	in-context few-shot	51.51	**85.10**	67.60	55.85	65.02
Qwen-plus	original	52.32	80.74	70.20	64.24	66.88
	CoT	**61.59**	83.21	66.23	**67.55**	**69.65**
	BAG	57.62	**85.11**	64.90	64.82	68.11
	in-context few-shot	52.32	82.01	**70.86**	59.60	66.20
Qwen-max	original	58.60	**89.53**	**68.08**	**69.33**	**71.39**
	CoT	59.20	82.79	64.72	61.99	67.18
	BAG	57.61	88.28	67.33	66.33	69.89
	in-context few-shot	**59.35**	87.78	64.59	63.84	68.89
GPT-4o	original	52.63	82.32	**71.32**	**67.92**	**68.55**
	CoT	**57.12**	84.90	67.53	62.18	67.93
	BAG	53.73	85.11	66.92	63.36	67.28
	in-context few-shot	56.52	**85.40**	66.10	64.91	68.23
Deepseek V3	original	54.97	83.79	**70.20**	**66.89**	**68.96**
	CoT	**59.60**	85.29	62.91	65.56	68.34
	BAG	54.77	**89.53**	64.24	56.95	66.37
	in-context few-shot	58.28	85.54	63.58	62.25	67.41
Gemini2.5 Pro	original	53.29	84.00	**70.98**	**68.52**	69.20
	CoT	60.10	84.13	68.62	66.40	**69.81**
	BAG	52.84	**87.02**	68.93	65.19	68.50
	in-context few-shot	**59.67**	85.62	66.35	63.92	68.89
Llama3B	1-hop w/o label	39.48	64.50	29.50	53.00	46.62
	2-hop w/o label	**49.63**	**69.92**	**29.51**	**56.10**	**51.28**
Llama8B	1-hop w/o label	58.35	73.07	61.85	59.85	63.28
	2-hop w/o label	**62.84**	**83.29**	**68.33**	**59.60**	**68.52**
Qwen-plus	1-hop w/o label	68.87	85.73	**73.83**	**72.19**	75.16
	2-hop w/o label	**76.16**	**88.98**	73.51	71.56	**77.55**
tuned Llama3B	ego	67.08	89.28	66.58	65.59	72.13
	1-hop w/o label	82.04	90.02	71.32	73.07	79.11
	2-hop w/o label	**85.04**	**91.52**	**72.82**	**77.89**	**81.82**
tuned Llama8B	ego	77.31	92.36	70.12	73.74	78.38
	1-hop w/o label	84.54	93.90	74.33	80.33	83.28
	2-hop w/o label	**89.67**	**95.22**	**76.01**	**84.51**	**86.35**

Table 16: Comparison of different LLMs on node classification. The bolded parts are used to compare the effects of using structural information and instruction tuning. The **best** results in each category are highlighted. The underline means the overall best result.

introduced by BAG. Smaller models may struggle with multi-step inference and instead rely on more direct input-output mappings. Constructing a graph before classification might exceed their reasoning capacity, leading to performance declines.

In-context few-shot prompting improves results on Cora and PubMed but underperforms on ArXiv and Products. Due to token limitations, only three example categories are included in the few-shot prompt. This coverage is insufficient for datasets with a large number of classes, making it difficult for the model to generalize to unseen categories.

Finally, incorporating structural information is more effective than using CoT, BAG, or in-context few-shot prompting for improving LLM performance. The greatest improvement comes from instruction tuning, as even smaller models with proper tuning can significantly outperform larger untuned models. However, the trade-off is the higher computational cost and longer training time required for instruction tuning.

Remark 7. *Larger models generally outperform smaller models in node classification tasks. CoT and in-context few-shot prompting significantly improve performance on small-class datasets, but may backfire on large-class datasets due to category ambiguity and token limitations. BAG imposes a heavy burden on smaller models, leading to noticeable performance drops. Instruction tuning combined with structural information yields the best results, though it requires careful consideration of computational costs.*

K Qualitative Analysis and Error Cases

To provide deeper insight beyond quantitative metrics, we performed a qualitative analysis of the instruction-tuned Llama8B model's predictions on the Cora dataset for the node classification task (using the 2-hop w/o label prompt). This analysis helps to illustrate both the strengths and weaknesses of the model.

K.1 Example of a Correct, Non-Trivial Prediction

Consider a target paper **P_target** with the title "Reinforcement Learning for Robot Soccer." Its neighbors include papers on "Multi-agent Learning" and "Q-Learning Applications." The model correctly classifies this paper under the **Reinforcement Learning** category.

Analysis: This case demonstrates the model's

strength in leveraging semantic understanding. While a traditional GNN would rely purely on the citation structure, the LLM effectively uses the textual information from the target node and its neighbors. The titles of the neighboring papers provide strong contextual clues that reinforce the classification, and the LLM successfully integrates this information to make a confident prediction. It reasons that "Robot Soccer" is a common application domain for "Multi-agent Learning," both of which are core topics within Reinforcement Learning.

K.2 Example of a Common Error Case: Over-reliance on Textual Cues

Consider a target paper **P_target** titled "A Probabilistic Framework for Genetic Sequence Analysis." Its 1-hop neighbors are primarily from the *Genetic Algorithms* class. However, **P_target** itself belongs to the *Probabilistic Methods* class, and its 2-hop neighborhood is more diverse. The model incorrectly classifies the paper as **Genetic Algorithms**.

Analysis: This is a classic example of where the model's strong textual priors can override structural information. The term "Genetic" in the title creates a strong semantic link to the *Genetic Algorithms* category. The model gives this textual signal more weight than the subtle structural clues that might point towards *Probabilistic Methods*. A classic GNN, immune to textual semantics, might have performed better in this specific case if the broader graph structure supported the correct class. This highlights a key challenge: teaching LLMs to balance textual information with graph topology, especially when they conflict.

K.3 Example of a Structural Reasoning Failure

Consider a target paper **P_target** whose textual content is ambiguous and could plausibly fit into two categories, e.g., *Neural Networks* and *Theory*. Its immediate 1-hop neighborhood is evenly split between these two classes. However, a large number of its 2-hop neighbors are strongly associated with the *Neural Networks* class. The model fails to make a decisive classification and often defaults to the more general *Theory* class or guesses incorrectly.

Analysis: This error suggests a potential limitation in the model's ability to effectively aggregate information from higher-order neighborhoods (2-hop and beyond) when the local signal is noisy or ambiguous. While the 2-hop information is provided in the prompt, the model may struggle to reason about the "weight of evidence" from these more distant nodes compared to the immediate neighbors. Improving the model's ability to reason over multi-hop information and recognize larger community structures remains an important direction for future work.

L Prompt Formats

L.1 Prompt Formats for Node Classification

As discussed in Section 3.1, there are five different prompt formats in node classification. We list them in Table 17 and describe them in detail.

L.2 Prompt Formats for Link Prediction

In Section 3.1, we design nine different prompt formats for link prediction, which are used for both instruction tuning and testing. These formats include:

1. **1-hop**: The task is to determine if there is an edge between target node1 and target node2. The prompt provides the 1-hop neighbors and their descriptions for both nodes.

2. **2-hop**: Similar to the 1-hop prompt but includes 2-hop neighbors and their descriptions for both target nodes.

3. **1-hop node judge**: Determine whether a specific node is a 1-hop neighbor of the target node.

4. **2-hop node judge**: Determine whether a specific node is a 2-hop neighbor of the target node.

5. **3-hop node judge**: Determine whether a specific node is a 3-hop neighbor of the target node.

6. **Middle node connection**: Determine if target node1 and target node2 are connected via a middle node.

7. **1-hop node fill-in**: Given the 1-hop neighbors of a target node, identify an additional node that is also a 1-hop neighbor.

8. **1-hop node selection**: Choose the correct 1-hop neighbor of the target node from four options (A, B, C, D).

9. **2-hop node selection**: Choose the correct 2-hop neighbor of the target node from four options (A, B, C, D).

To ensure the reasoning is non-trivial, target node1 and target node2 must not appear in each other's 1-hop or 2-hop neighborhoods. Table 18 provides a detailed description of these nine prompt formats.

L.3 Prompt Formats for Pure Graph Structure

In Section J.1, we propose removing all node attributes and keeping only node IDs to focus solely on the structural reasoning capabilities of LLMs. We refer to these graph prompts as "prompts for pure graph structure". In Table 19, we use the **1-hop w/o label** prompts for node classification and **1-hop** prompts for link prediction as examples, as the logic for other prompt formats follows a similar approach.

M Use of Large Language Models

During the preparation of this manuscript, a Large Language Model (LLM) was utilized as a writing aid to improve the overall linguistic quality and clarity. This assistance was confined to copy-editing tasks, such as correcting grammatical and spelling errors, rephrasing sentences for enhanced flow and readability, and ensuring conciseness. All scientific contributions, including the research ideas, experimental design, analysis, and conclusions presented herein, are entirely the original work of the human authors.

Prompt Formats	Description
ego	**"Context"**: "You are a good graph reasoner. Given a graph language that describes the target node information from the Cora dataset, you need to understand the graph and the task definition and answer the question. (<Target node>, <Node attributes>)", **"Question"**: "Please predict the most appropriate category for the Target node. Choose from the following categories: <Categories>. Do not provide your reasoning. Answer: ", **"Answer"**: "<Correct answer>" **Example:** (<Target node>, <Node attributes>): ## Target node: \nPaper id: 540 \nTitle: A Model-Based Approach to Blame-Assignment in Design <Categories>: Rule Learning \nNeural Networks \nCase Based \nGenetic Algorithms \nTheory \nReinforcement Learning \nProbabilistic Methods <Correct answer>: Case Based
1-hop w/o label	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and the task definition and answer the question. (<Target node>, <Node attributes>), (<1-hop neighbors>, <Node attributes>)", **"Question"**: "Please predict the most appropriate category for the Target node. Choose from the following categories: <Categories>. Do not provide your reasoning. Answer: ", **"Answer"**: "<Correct answer>" (<Target node>, <Node attributes>): ## Target node: \nPaper id: 197 \nTitle: Optimal Navigation in a Probibalistic World (<1-hop neighbors>, <Node attributes>): Known neighbor papers at hop 1 (partial, may be incomplete): \nPaper id: 295 \nTitle: A Neuro-Dynamic Programming Approach to Retailer Inventory Management 1 \nPaper id: 749 \nTitle: On the Complexity of Solving Markov Decision Problems \nPaper id: 3 \nTitle: Planning and Acting in Partially Observable Stochastic Domains \nPaper id: 633 \nTitle: Chapter 1 Reinforcement Learning for Planning and Control <Categories>: Rule Learning \nNeural Networks \nCase Based \nGenetic Algorithms \nTheory \nReinforcement Learning \nProbabilistic Methods <Correct answer>: Reinforcement Learning

Prompt Formats	Description
2-hop w/o label	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and the task definition and answer the question. (<Target node>, <Node attributes>), (<1-hop neighbors>, <Node attributes>), (<2-hop neighbors>, <Node attributes>)", **"Question"**: "Please predict the most appropriate category for the Target node. Choose from the following categories: <Categories>. Do not provide your reasoning. Answer: ", **"Answer"**: "<Correct answer>" (<Target node>, <Node attributes>): ## Target node: \nPaper id: 546 \nTitle: GREQE a Diplome des Etudes Approfondies en Economie Mathematique et Econometrie (<1-hop neighbors>, <Node attributes>): Known neighbor papers at hop 1 (partial, may be incomplete): \nPaper id: 163 \nTitle: 4 Implementing Application Specific Routines Genetic algorithms in search, optimization, and machine learning (<2-hop neighbors>, <Node attributes>): Known neighbor papers at hop 2 (partial, may be incomplete): \nPaper id: 1573 \nTitle: Genetics-based Machine Learning and Behaviour Based Robotics: A New Synthesis complexity grows \nPaper id: 1069 \nTitle: Extended Selection Mechanisms in Genetic Algorithms \nPaper id: 2232 \nTitle: Facing The Facts: Necessary Requirements For The Artificial Evolution of Complex Behaviour <Categories>: Rule Learning \nNeural Networks \nCase Based \nGenetic Algorithms \nTheory \nReinforcement Learning \nProbabilistic Methods <Correct answer>: Genetic Algorithms
1-hop w label	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and the task definition and answer the question. (<Target node>, <Node attributes>), (<1-hop neighbors>, <Node attributes>, <Labels>)", **"Question"**: "Please predict the most appropriate category for the Target node. Choose from the following categories: <Categories>. Do not provide your reasoning. Answer: ", **"Answer"**: "<Correct answer>" (<Target node>, <Node attributes>): ## Target node: \nPaper id: 2156 \nTitle: WORST CASE PREDICTION OVER SEQUENCES UNDER LOG LOSS (<1-hop neighbors>, <Node attributes>, <Labels>): Known neighbor papers at hop 1 (partial, may be incomplete): \nPaper id: 2098 \nTitle: Predicting a binary sequence almost as well as the optimal biased coin \nLabel: Theory \nPaper id: 453 \nTitle: How to Use Expert Advice (Extended Abstract) \nLabel: Theory <Categories>: Rule Learning \nNeural Networks \nCase Based \nGenetic Algorithms \nTheory \nReinforcement Learning \nProbabilistic Methods <Correct answer>: Theory

Prompt Formats	Description
2-hop w label	"Context": "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and the task definition and answer the question. (<Target node>, <Node attributes>), (<1-hop neighbors>, <Node attributes>, <Labels>), (<2-hop neighbors>, <Node attributes>, <Labels>)", "Question": "Please predict the most appropriate category for the Target node. Choose from the following categories: <Categories>. Do not provide your reasoning. Answer: ", "Answer": "<Correct answer>" (<Target node>, <Node attributes>): ## Target node: \nPaper id: 1443 \nTitle: Residual Q-Learning Applied to Visual Attention (<1-hop neighbors>, <Node attributes>, <Labels>): Known neighbor papers at hop 1 (partial, may be incomplete): \nPaper id: 1540 \nTitle: MultiPlayer Residual Advantage Learning With General Function Approximation \nPaper id: 1540 \nTitle: MultiPlayer Residual Advantage Learning With General Function Approximation (<2-hop neighbors>, <Node attributes>, <Labels>): Known neighbor papers at hop 2 (partial, may be incomplete): \nPaper id: 565 \nTitle: Machine Learning Learning to Predict by the Methods of Temporal Differences Keywords \nLabel: Reinforcement Learning \nPaper id: 842 \nTitle: Metrics for Temporal Difference Learning <Categories>: Rule Learning \nNeural Networks \nCase Based \nGenetic Algorithms \nTheory \nReinforcement Learning \nProbabilistic Methods <Correct answer>: Reinforcement Learning

Table 17: Prompt formats for node classification.

Prompt Formats	Description
1-hop	"Context": "You are a good graph reasoner. Based on the cora dataset, determine whether two target nodes are connected by an edge. When you make a decision, please carefully consider the graph structure and the node information. If two nodes share similar structure or information, they are likely to be connected. (<Target node1>, <Node attributes>), (<1-hop neighbors>, <Node attributes>), (<Target node2>, <Node attributes>), (<1-hop neighbors>, <Node attributes>)", "Question": "Are Target Node1 and Target Node2 connected? Do not provide your reasoning. Only provide "Yes" or "No" based on your inference. Answer: ", "Answer": "<Correct answer>"
2-hop	"Context": "You are a good graph reasoner. Based on the cora dataset, determine whether two target nodes are connected by an edge. When you make a decision, please carefully consider the graph structure and the node information. If two nodes share similar structure or information, they are likely to be connected. (<Target node1>, <Node attributes>), (<1-hop neighbors>, <Node attributes>), (<2-hop neighbors>, <Node attributes>), (<Target node2>, <Node attributes>), (<1-hop neighbors>, <Node attributes>), (<2-hop neighbors>, <Node attributes>)", "Question": "Are Target Node1 and Target Node2 connected? Do not provide your reasoning. Only provide "Yes" or "No" based on your inference. Answer: ", "Answer": "<Correct answer>"

Prompt Formats	Description
1-hop node judge	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and answer the question. When you make a decision, please carefully consider the graph structure and the node information. (<Target node1>, <Node attributes>), (<1-hop neighbors>, <Node attributes>), (<Target node2>, <Node attributes>), (<1-hop neighbors>, <Node attributes>)", **"Question"**: "Based on the available partial information. Are Target Node1 and Target Node2 connected? Do not provide your reasoning. Only provide "Yes" or "No" based on your inference. Answer: ", **"Answer"**: "<Correct answer>"
2-hop node judge	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and answer the question. When you make a decision, please carefully consider the graph structure and the node information. (<Target node1>, <Node attributes>), (<1-hop neighbors>, <Node attributes>), (<2-hop neighbors>, <Node attributes>), (<Target node2>, <Node attributes>)", **"Question"**: "Based on the available partial information. Can Target node2 be a 2-hop neighbor of Target node1? Do not provide your reasoning. Only provide "Yes" or "No" based on your inference. Answer: ", **"Answer"**: "<Correct answer>"
3-hop node judge	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and answer the question. When you make a decision, please carefully consider the graph structure and the node information. (<Target node1>, <Node attributes>), (<1-hop neighbors>, <Node attributes>), (<2-hop neighbors>, <Node attributes>), (<Target node2>, <Node attributes>), (<1-hop neighbors>, <Node attributes>)", **"Question"**: "Based on the available partial information. Can Target node2 be a 3-hop neighbor of Target node1? Do not provide your reasoning. Only provide "Yes" or "No" based on your inference. Answer: ", **"Answer"**: "<Correct answer>"
Middle node connection	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and answer the question. When you make a decision, please carefully consider the graph structure and the node information. (<Target node1>, <Node attributes>), (<1-hop neighbors>, <Node attributes>), (<Target node2>, <Node attributes>), (<Middle node>, <Node attributes>)", **"Question"**: "Can Target node1 be connected with Target node2 through the Middle node? Do not provide your reasoning. Only provide "Yes" or "No" based on your inference. Answer: ", **"Answer"**: "<Correct answer>"
1-hop node fill-in	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and answer the question. When you make a decision, please carefully consider the graph structure and the node information. (<Target node1>, <Node attributes>), (<1-hop neighbors>, <Node attributes>)", **"Question"**: "Based on the available partial information. Which other node will be connected to Target node1 within one hop? Do not provide your reasoning. The answer should be the paper id. Answer: ", **"Answer"**: "<Correct answer>"

Prompt Formats	Description
1-hop node selection	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and answer the question. When you make a decision, please carefully consider the graph structure and the node information. (<Target node1>, <Node attributes>), (<1-hop neighbors>, <Node attributes>)", **"Question"**: "Based on the available partial information. Which other node can be connected to Target node1 within one hop? A.(<Node A>,<Attribute>) \nB.(<Node B>,<Attribute>) \nC.(<Node C>,<Attribute>) \nD.(<Node D>,<Attribute>) Do not provide your reasoning. The answer should be A, B, C or D. Answer: ", **"Answer"**: "<Correct answer>"
2-hop node selection	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and answer the question. When you make a decision, please carefully consider the graph structure and the node information. (<Target node1>, <Node attributes>), (<1-hop neighbors>, <Node attributes>), (<2-hop neighbors>, <Node attributes>)", **"Question"**: "Based on the available partial information. Which other node can be a 2-hop neighbor of Target node1? A.(<Node A>,<Attribute>) \nB.(<Node B>,<Attribute>) \nC.(<Node C>,<Attribute>) \nD.(<Node D>,<Attribute>) Do not provide your reasoning. The answer should be A, B, C or D. Answer: ", **"Answer"**: "<Correct answer>"

Table 18: Prompt formats for link prediction.

Prompt Formats	Description
1-hop w/o label (Node classification)	**"Context"**: "You are a good graph reasoner. Give you a graph language that describes a graph structure and node information from cora dataset. You need to understand the graph and the task definition and answer the question. <Target node>, <1-hop neighbors>", **"Question"**: "Please predict the most appropriate category for the Target node. Choose from the following categories: <Categories>. Do not provide your reasoning. Answer: ", **"Answer"**: "<Correct answer>" (<Target node>, <Node attributes>): ## Target node: \nPaper id: 197 <1-hop neighbors>: Known neighbor papers at hop 1 (partial, may be incomplete): \nPaper id: 295 \nPaper id: 749 \nPaper id: 3 \nPaper id: 633 <Categories>: Rule Learning \nNeural Networks \nCase Based \nGenetic Algorithms \nTheory \nReinforcement Learning \nProbabilistic Methods <Correct answer>: Reinforcement Learning

Prompt Formats	Description
1-hop (Link prediction)	**"Context"**: "You are a good graph reasoner. Based on the cora dataset, determine whether two target nodes are connected by an edge. When you make a decision, please carefully consider the graph structure and the node information. If two nodes share similar structure or information, they are likely to be connected. <Target node1>, <1-hop neighbors>, <Target node2>, <1-hop neighbors>", **"Question"**: "Are Target Node1 and Target Node2 connected? Do not provide your reasoning. Only provide "Yes" or "No" based on your inference. Answer: ", **"Answer"**: "<Correct answer>" **Example:** <Target node1>: ## Target node1: \nPaper id: 172 <1-hop neighbors>: Known neighbor papers at hop 1 (partial, may be incomplete): \nPaper id: 635 \nPaper id: 430 <Target node2>: ## Target node2: \nPaper id: 245 <1-hop neighbors>: Known neighbor papers at hop 1 (partial, may be incomplete): \nPaper id: 1636 <Correct answer>: Yes

Table 19: Prompt formats for pure graph structure.