A Comprehensive Analysis on LLM-based Node Classification Algorithms

Xixi Wu [1] **Yifei Shen** [2] **Fangzhou Ge** [1] **Caihua Shan** [2] **Yizhu Jiao** [3] **Xiangguo Sun** [1] **Hong Cheng** [1]

Abstract

Node classification is a fundamental task in graph analysis, with broad applications across various fields. Recent breakthroughs in Large Language Models (LLMs) have enabled LLM-based approaches for this task. Although many studies demonstrate the impressive performance of LLM-based methods, the lack of clear design guidelines may hinder their practical application. In this work, we aim to establish such guidelines through a fair and systematic comparison of these algorithms. As a first step, we developed LLMNodeBed, a comprehensive codebase and testbed for node classification using LLMs. It includes ten datasets, eight LLM-based algorithms, and three learning paradigms, and is designed for easy extension with new methods and datasets. Subsequently, we conducted extensive experiments, training and evaluating over 2,200 models, to determine the key settings (e.g., learning paradigms and homophily) and components (e.g., model size) that affect performance. Our findings uncover eight insights, e.g., (1) LLM-based methods can significantly outperform traditional methods in a semi-supervised setting, while the advantage is marginal in a supervised setting; (2) Graph Foundation Models can beat open-source LLMs but still fall short of strong LLMs like GPT-4o in a zero-shot setting. We hope that the release of LLMNodeBed, along with our insights, will facilitate reproducible research and inspire future studies in this field[1].

1. Introduction

Node classification is a fundamental task in graph analysis, with a wide range of applications such as item tagging (Mao et al., 2020), user profiling (Yan et al., 2021), and financial fraud detection (Zhang et al., 2022a). Developing effective algorithms for node classification is crucial, as they can significantly impact commercial success. For instance, US banks lost 6 billion USD to fraudsters in 2016. Therefore, even a marginal improvement in fraud detection accuracy could result in substantial financial savings.

Given its practical importance, node classification has been a long-standing research focus in both academia and industry. The earliest attempts to address this task adopted techniques such as Laplacian regularization (Belkin et al., 2006), graph embeddings (Yang et al., 2016a), and label propagation (Zhu et al., 2003). Over the past decade, GNN-based methods have been developed and have quickly become prominent due to their superior performance, as demonstrated by works such as Kipf & Welling (2017), Veličković et al. (2018), and Hamilton et al. (2017). Additionally, the incorporation of encoded textual information has been shown to further complement GNNs' node features, enhancing their effectiveness (Jin et al., 2023b; Zhao et al., 2023a).

Inspired by the recent success of LLMs, there has been a surge of interest in leveraging LLMs for node classification (Li et al., 2023). LLMs, pre-trained on extensive text corpora, possess context-aware knowledge and superior semantic comprehension, overcoming the limitations of the non-contextualized shallow embeddings used by traditional GNNs. Typically, supervised methods fall into three categories: Encoder, Reasoner, and Predictor. In the Encoder paradigm, LLMs employ their vast parameters to encode nodes' textual information, producing more expressive features that surpass shallow embeddings (Zhu et al., 2024b). The Reasoner approach utilizes LLMs' reasoning capabilities to enhance node attributes and the task descriptions with a more detailed text (Chen et al., 2024b; He et al., 2023). This generated text augments the nodes' original information, thereby enriching their attributes. Lastly, the Predictor role involves LLMs integrating graph context through graph encoders, enabling direct text-based predictions (Chen et al., 2024a; Tang et al., 2023; Chai et al., 2023; Huang et al., 2024). For zero-shot learning with LLMs, methods can be categorized into two types: Direct Inference and Graph Foundation Models (GFMs). Direct Inference involves guiding LLMs to directly perform classification tasks via crafted prompts (Huang et al., 2023). In contrast, GFMs entail

[1]The Chinese University of Hong Kong [2]Microsoft Research Asia [3]University of Illinois Urbana-Champaign. Correspondence to: Yifei Shen <yifeishen@microsoft.com>, Hong Cheng <hcheng@se.cuhk.edu.hk>.

[1]Codes and datasets of LLMNodeBed are released at https://github.com/WxxShirley/LLMNodeBed

pre-training on extensive graph corpora before applying the model to target graphs, thereby equipping the model with specialized graph intelligence (Li et al., 2024b). An illustration of these methods is shown in Figure 2.

Despite tremendous efforts and promising results, the design principles for LLM-based node classification algorithms remain elusive. Given the significant training and inference costs associated with LLMs, practitioners may opt to deploy these algorithms only when they provide substantial performance enhancements compared to costs. This study, therefore, seeks to identify **(1) the most suitable settings for each algorithm category, and (2) the scenarios where LLMs surpass traditional LMs such as BERT**. While recent work like GLBench (Li et al., 2024c) has evaluated various methods using consistent data splits in semi-supervised and zero-shot settings, differences in backbone architectures and implementation codebases still hinder fair comparisons and rigorous conclusions. To address these limitations, we introduce a new benchmark that further standardizes backbones and codebases. Additionally, we extend GLBench by incorporating three new E-Commerce datasets relevant to practical applications and expanding the evaluation settings. Specifically, we assess the impact of supervision signals (e.g., supervised, semi-supervised), different language model backbones (e.g., RoBERTa, Mistral, LLaMA, GPT-4o), and various prompt types (e.g., CoT, ToT, ReAct). These enhancements enable a more detailed and reliable analysis of LLM-based node classification methods. In summary, our contributions to the field of LLMs for graph analysis are as follows:

- **A Testbed:** We release LLMNodeBed, a PyG-based testbed designed to facilitate reproducible and rigorous research in LLM-based node classification algorithms. The initial release includes ten datasets, eight LLM-based algorithms, and three learning configurations. LLMNodeBed allows for easy addition of new algorithms or datasets, and a single command to run all experiments, and to automatically generate all tables included in this work.
- **Comprehensive Experiments:** By training and evaluating over 2,200 models, we analyzed how the learning paradigm, homophily, language model type and size, and prompt design impact the performance of each algorithm category.
- **Insights and Tips:** Detailed experiments were conducted to analyze each influencing factor. We identified the settings where each algorithm category performs best and the key components for achieving this performance. Our work provides intuitive explanations, practical tips, and insights about the strengths and limitations of each algorithm category.

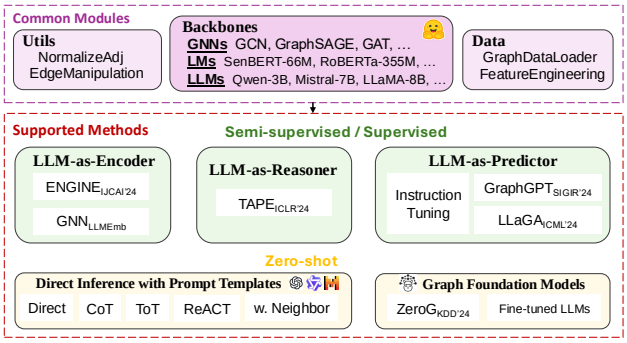


Figure 1: **Overview of LLMNodeBed.**

2. Preliminaries on Node Classification

To leverage the language abilities of LLMs, we study the node classification task within the context of text-attributed graphs (TAGs) (Ma & Tang, 2021). TAGs are represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{S})$, where $\mathcal{V}$ represents the set of nodes, $\mathcal{E}$ the set of edges, and $\mathcal{S}$ the collection of textual descriptions associated with each node $v \in \mathcal{V}$. Some of the nodes are associated with labels, represented as $\mathcal{V}_l \subset \mathcal{V}$. The remaining nodes do not have labels, and are denoted as $\mathcal{V}_u$. The goal of node classification is to train a neural network based on the graph $\mathcal{G}$ and the labels of $\mathcal{V}_l$, which can predict the labels of the unlabeled nodes in $\mathcal{V}_u$.

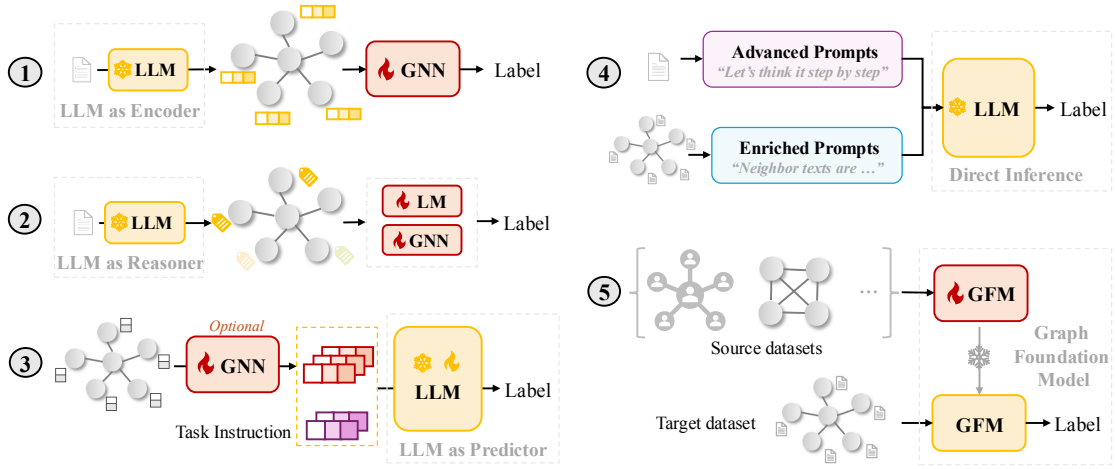
Traditionally, the textual attributes of nodes can be encoded into shallow embeddings as $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{|\mathcal{V}|}] \in \mathbb{R}^{|\mathcal{V}| \times d}$ using naive methods like bag-of-words or TF-IDF (Salton & Buckley, 1988), where d represents the dimensionality of the embeddings. Such transformation is adopted in most GNN papers. Instead, the input for LLM-based approaches is the raw text and one may expect that the pre-trained knowledge in LLMs can improve performance.

3. LLMNodeBed: A Testbed for LLM-based Node Classification

In this section, we present the datasets, baselines, and learning paradigms within LLMNodeBed (Figure 1).

3.1. Datasets

To provide guidelines for applying algorithms across diverse real-world applications, the selection of datasets in LLMNodeBed considers several key factors: (1) **Multi-domain Diversity** to reflect different contexts, (2) **Varying Scales** to examine algorithm scalability and the associated costs of leveraging LLMs, and (3) **Diverse Levels of Homophily** to understand its impact on performance. Therefore, LLMNodeBed comprises ten datasets spanning the academic, web link, social, and E-Commerce domains. These datasets vary significantly in scale, ranging from thou-



(a) Semi- / Supervised Settings **(b) Zero-shot Setting**

Figure 2: **Illustrations of LLM-based node classification algorithms under supervised and zero-shot settings.**

sands of nodes to millions of edges, and exhibit differing levels of homophily. Such diversity in domain, scale, and homophily enables the assessment of algorithms across a wide range of contexts.

For datasets where raw text has been preprocessed into vector embeddings using bag-of-words or TF-IDF techniques, we utilize collected versions including Cora and Pubmed (He et al., 2023), Citeseer (Chen et al., 2024b), and WikiCS (Liu et al., 2024b). The remaining datasets already include text attributes in their official releases, including arXiv (Hu et al., 2020), Instagram and Reddit (Huang et al., 2024), and Books, Computer, and Photo (Yan et al., 2023). Detailed statistics and information for these datasets are provided in Table 8 and Table 9 in the Appendix.

3.2. Baselines

The initial release of LLMNodeBed includes eight LLM-based baseline algorithms alongside classic methods. We selected these LLM-based algorithms based on three key criteria: (1) **Diverse Roles of LLMs** to thoroughly evaluate their effectiveness, (2) **Straightforward Design** to facilitate clear comparisons by avoiding complex and intertwined architectures, and (3) **Representativeness** to ensure benchmark relevance by including widely recognized methods. Therefore, the LLM-based baselines include:

LLM-as-Encoder: We include **ENGINE** (Zhu et al., 2024b) and introduce **GNN$_{\text{LLMEmb}}$**. ENGINE aggregates hidden embeddings from each LLM layer to create comprehensive node representations. In contrast, GNN$_{\text{LLMEmb}}$ initializes node embeddings using the LLM's last hidden layer before feeding them into GNNs for classification.

LLM-as-Reasoner: We select **TAPE** (He et al., 2023), a representative LLM-as-Reasoner method. TAPE prompts the LLM to reason over nodes by generating predictions along with explanations, thereby enriching the node's text attributes and enhancing classification performance.

Both Encoder and Reasoner methods require processing the entire dataset, either by encoding each node's text or by performing reasoning over nodes, which introduces additional processing time before the actual model training begins.

LLM-as-Predictor: We select **GraphGPT** (Tang et al., 2023), **LLaGA** (Chen et al., 2024a), and implement **LLM Instruction Tuning** (**LLM$_{\text{IT}}$**). GraphGPT employs a multi-stage pre-training and instruction tuning process to classify nodes based on both text and graph context. **LLaGA** integrates tokenized task instructions and graph context into LLMs to generate predictions. We implement LLM$_{\text{IT}}$ to evaluate whether LLMs alone can function as effective predictors. This involves fine-tuning the LLM using task prompts and ground-truth labels formatted as $\langle$`Question,Answer`$\rangle$ pairs.

LLM Direct Inference: This category refers to LLMs generating prediction labels directly from a node's text without additional training or labels. We employ two types of prompt templates: (1) **Advanced Prompts** that improve LLMs' reasoning abilities such as Chain-of-Thought (CoT) (Wei et al., 2022) and Tree-of-Thought (ToT) (Yao et al., 2023a), and (2) **Enriched Prompts** that incorporate neighboring node information to provide structural context.

GFMs: GFMs are foundation models trained on large-scale source graph datasets to acquire general classification knowledge, which can then be seamlessly applied to target graphs. We include **ZeroG** (Li et al., 2024b) as a representative GFM due to its superior performance in zero-shot settings. Additionally, LLM-as-Predictor methods trained with extensive graph corpora are also considered within this

category for zero-shot applications.

Besides LLM-based methods, LLMNodeBed also integrates classic algorithms, including **MLPs**, **GNNs**, and **LMs**. MLPs generate predicted label matrices from node embeddings, while GNNs combine shallow embeddings with graph structures for label prediction. LMs process a node's text through hidden layers and use a classification head to produce label distributions. Further discussions of existing node classification algorithms are provided in Appendix A.

Prompt templates for LLM-as-Predictor and Direct Inference are listed in Appendix B.1 and B.2, respectively. Appendix C.2 describes the implementation details, backbone selections, and hyperparameter search spaces for all algorithms. Additionally, we highlight the distinctions of LLMNodeBed in Appendix C.3.

3.3. Learning Paradigms

We evaluate the baselines under three learning configurations: Semi-supervised, Supervised, and Zero-shot. These configurations are defined as follows:

- **Semi-supervised Learning:** A small subset of nodes $\mathcal{V}_l \subseteq \mathcal{V}$ with known labels $\mathcal{Y}_l$ is provided. This setting assesses the model's ability to effectively utilize limited labeled data, reflecting real-world scenarios where labeling is scarce. For experimental datasets, we adopt the official splits designed for semi-supervised settings to ensure standardized evaluation.
- **Supervised Learning:** A larger subset of nodes $\mathcal{V}_l$ with known labels is provided, assessing the model's performance with abundant supervision. Specifically, we use a 60% training, 20% validation, and 20% testing split for most datasets. This consistent split facilitates fair comparisons across baselines. Detailed data splits are provided in Table 9 in the Appendix.
- **Zero-shot Learning:** No labeled data is provided for training. The model predicts labels solely based on node textual descriptions and the graph structure, assessing its ability to generalize to new, unseen data. For test samples, we follow existing literature (Zhu et al., 2024a) by selecting one smaller dataset from each domain and using 20% of its nodes as test samples.

4. Comparisons among Algorithm Categories

In this section, we present empirical results among algorithm categories, along with key insights derived from them.

4.1. Semi-supervised and Supervised Performance

Settings: To ensure a fair comparison of baseline algorithms, all methods are implemented with consistent components: GCN (Kipf & Welling, 2017) for GNNs, RoBERTa-355M (Liu et al., 2019) for LMs, and Mistral-7B (Jiang et al., 2023) for LLM components where applicable. This uniformity guarantees that performance differences are attributable to model designs rather than underlying architectures. Each experiment was conducted over **4 runs**. Based on the results of Accuracy (Table 1) and Macro-F1 (Table 10 in Appendix), we summarize the following takeaways:

Takeaway 1: Appropriately incorporating LLMs consistently improves the performance. According to the table, the best performance is often achieved by LLM-based methods compared to classic methods. It suggests that using LLM to exploit the textual information is useful.

Takeaway 2: LLM-based methods provide greater improvements in semi-supervised settings than in supervised settings. By comparing the tables, we observe that performance gains are more significant in semi-supervised scenarios. From an information-theoretic perspective, the node classification task with cross-entropy loss aims to maximize the mutual information between the graph and the provided labels, denoted as $I(\mathcal{G}; \mathcal{Y}_l)$. If we consider graph $\mathcal{G}$ as a joint distribution of node attributes $\boldsymbol{X}$ and structure $\mathcal{E}$, we have:

$$I(\mathcal{G}; \mathcal{Y}_l) = I(\boldsymbol{X}, \mathcal{E}; \mathcal{Y}_l) = I(\mathcal{E}; \mathcal{Y}_l) + I(\boldsymbol{X}; \mathcal{Y}_l | \mathcal{E}). \quad (1)$$

The first term represents the information encoded in the graph structure, utilized by classic GNNs, while the second term represents information from node features, leveraged by LLMs. In semi-supervised settings, the mutual information between structure and labels is relatively low, allowing LLMs to contribute more significantly to performance.

Takeaway 3: LLM-as-Reasoner methods are highly effective when labels heavily depend on text. TAPE achieves top or runner-up performance on academic and web link datasets like Cora and WikiCS, where structural information is less relevant to labels (Zhang et al., 2022b). However, TAPE struggles with social networks that require deeper structural understanding, such as predicting popular users (high-degree nodes) on Reddit.

Takeaway 4: LLM-as-Encoder methods balance computational cost and accuracy effectively. LLM-as-Encoder methods perform satisfactorily across all datasets. Further experiments in Section 5.1 reveal that **LLM-as-Encoder methods are more effective than their LM counterparts when graphs are heterophilic.** Regarding cost-effectiveness, LLM-as-Reasoner should generate long reasoning text, which is far more time-consuming than encoding texts in LLM-as-Encoder (see Appendix F). Therefore, LLM-as-Encoder methods strike a balance between computational efficiency and accuracy.

Takeaway 5: LLM-as-Predictor methods are more effec-

Table 1: **Performance comparison under semi-supervised and supervised settings with Accuracy (%) reported.** The **best** and second-best results are highlighted. Results of Macro-F1 are shown in Table 10 in the Appendix. LLM_{IT} on the arXiv dataset requires 30+ hours per run, preventing multiple executions.

Semi-supervised		Cora	Citeseer	Pubmed	WikiCS	Instagram	Reddit	Books	Photo	Computer	Avg.
Classic	$\text{GCN}_{\text{ShallowEmb}}$	$82.30_{\pm 0.19}$	$70.55_{\pm 0.32}$	$78.94_{\pm 0.27}$	$79.86_{\pm 0.19}$	$63.50_{\pm 0.11}$	$61.44_{\pm 0.38}$	$68.79_{\pm 0.46}$	$69.25_{\pm 0.81}$	$71.44_{\pm 1.19}$	71.79
	$\text{SAGE}_{\text{ShallowEmb}}$	$82.27_{\pm 0.37}$	$69.56_{\pm 0.43}$	$77.88_{\pm 0.44}$	$79.67_{\pm 0.25}$	$63.57_{\pm 0.10}$	$56.65_{\pm 0.33}$	$72.01_{\pm 0.33}$	$78.50_{\pm 0.15}$	$81.43_{\pm 0.27}$	73.50
	$\text{GAT}_{\text{ShallowEmb}}$	$81.30_{\pm 0.67}$	$69.94_{\pm 0.74}$	$78.49_{\pm 0.70}$	$79.99_{\pm 0.65}$	$63.56_{\pm 0.04}$	$60.60_{\pm 1.17}$	$74.35_{\pm 0.35}$	$80.40_{\pm 0.45}$	$83.39_{\pm 0.22}$	74.67
	SenBERT-66M	$66.66_{\pm 1.42}$	$60.52_{\pm 1.62}$	$36.04_{\pm 2.92}$	$77.77_{\pm 0.75}$	$59.00_{\pm 1.17}$	$56.05_{\pm 0.41}$	$83.68_{\pm 0.19}$	$73.89_{\pm 0.31}$	$70.76_{\pm 0.15}$	64.93
	RoBERTa-355M	$72.24_{\pm 1.14}$	$66.68_{\pm 2.03}$	$42.32_{\pm 1.56}$	$76.81_{\pm 1.04}$	$63.52_{\pm 0.44}$	$59.27_{\pm 0.34}$	$84.62_{\pm 0.16}$	$74.79_{\pm 1.13}$	$72.31_{\pm 0.37}$	68.06
Encoder	$\text{GCN}_{\text{LLMEmb}}$	$83.33_{\pm 0.75}$	$71.39_{\pm 0.90}$	$78.71_{\pm 0.45}$	$80.94_{\pm 0.16}$	**$67.49_{\pm 0.43}$**	$68.65_{\pm 0.75}$	$83.03_{\pm 0.34}$	$84.84_{\pm 0.47}$	$88.22_{\pm 0.16}$	**78.51**
	ENGINE	**$84.22_{\pm 0.46}$**	**$72.14_{\pm 0.74}$**	$77.84_{\pm 0.27}$	$80.94_{\pm 0.19}$	$67.14_{\pm 0.46}$	$69.67_{\pm 0.16}$	$82.89_{\pm 0.14}$	$84.33_{\pm 0.57}$	$86.42_{\pm 0.23}$	78.40
Reasoner	TAPE	$84.04_{\pm 0.24}$	$71.87_{\pm 0.35}$	$78.61_{\pm 1.23}$	**$81.94_{\pm 0.16}$**	$66.07_{\pm 0.10}$	$62.43_{\pm 0.47}$	**$84.92_{\pm 0.26}$**	**$86.46_{\pm 0.12}$**	**$89.52_{\pm 0.04}$**	78.43
Predictor	LLM_{IT}	$67.00_{\pm 0.16}$	$54.26_{\pm 0.22}$	**$80.99_{\pm 0.43}$**	$75.02_{\pm 0.16}$	$41.83_{\pm 0.47}$	$54.09_{\pm 1.02}$	$80.92_{\pm 1.38}$	$71.28_{\pm 1.81}$	$66.99_{\pm 2.02}$	65.76
	GraphGPT	$64.72_{\pm 1.50}$	$64.58_{\pm 1.55}$	$70.34_{\pm 2.27}$	$70.71_{\pm 0.37}$	$62.88_{\pm 2.14}$	$58.25_{\pm 0.37}$	$81.13_{\pm 1.52}$	$77.48_{\pm 0.78}$	$80.10_{\pm 0.76}$	70.02
	LLaGA	$78.94_{\pm 1.14}$	$62.61_{\pm 3.63}$	$65.91_{\pm 2.09}$	$76.47_{\pm 2.20}$	$65.84_{\pm 0.72}$	**$70.10_{\pm 0.38}$**	$83.47_{\pm 0.45}$	$84.44_{\pm 0.90}$	$87.82_{\pm 0.53}$	75.07

Supervised		Cora	Citeseer	Pubmed	arXiv	WikiCS	Instagram	Reddit	Books	Photo	Computer	Avg.
Classic	$\text{GCN}_{\text{ShallowEmb}}$	$87.41_{\pm 2.08}$	$75.74_{\pm 1.20}$	$89.01_{\pm 0.59}$	$71.39_{\pm 0.28}$	$83.67_{\pm 0.63}$	$63.94_{\pm 0.61}$	$65.07_{\pm 0.38}$	$76.94_{\pm 0.26}$	$73.34_{\pm 1.34}$	$77.16_{\pm 3.80}$	76.37
	$\text{SAGE}_{\text{ShallowEmb}}$	$87.44_{\pm 1.74}$	$74.96_{\pm 1.20}$	$90.47_{\pm 0.25}$	$71.21_{\pm 0.18}$	$84.86_{\pm 0.91}$	$64.14_{\pm 0.47}$	$61.52_{\pm 0.60}$	$79.40_{\pm 0.45}$	$84.59_{\pm 0.32}$	$87.77_{\pm 0.34}$	78.64
	$\text{GAT}_{\text{ShallowEmb}}$	$86.68_{\pm 1.12}$	$73.73_{\pm 0.94}$	$88.25_{\pm 0.47}$	$71.57_{\pm 0.25}$	$83.94_{\pm 0.61}$	$64.93_{\pm 0.75}$	$64.16_{\pm 1.05}$	$80.61_{\pm 0.49}$	$84.84_{\pm 0.69}$	$88.32_{\pm 0.24}$	78.70
	SenBERT-66M	$79.61_{\pm 1.40}$	$74.06_{\pm 1.26}$	$94.47_{\pm 0.33}$	$72.66_{\pm 0.24}$	$86.51_{\pm 0.86}$	$60.11_{\pm 0.93}$	$58.70_{\pm 0.54}$	$85.99_{\pm 0.58}$	$77.72_{\pm 0.35}$	$74.22_{\pm 0.21}$	76.40
	RoBERTa-355M	$83.17_{\pm 0.84}$	$75.90_{\pm 1.69}$	**$94.84_{\pm 0.06}$**	$74.12_{\pm 0.12}$	**$87.47_{\pm 0.83}$**	$63.75_{\pm 1.13}$	$60.61_{\pm 0.24}$	**$86.65_{\pm 0.38}$**	$79.45_{\pm 0.37}$	$75.76_{\pm 0.30}$	78.17
Encoder	$\text{GCN}_{\text{LLMEmb}}$	**$88.15_{\pm 1.79}$**	$76.45_{\pm 1.19}$	$88.38_{\pm 0.68}$	$74.39_{\pm 0.31}$	$84.78_{\pm 0.86}$	$68.27_{\pm 0.45}$	$70.65_{\pm 0.75}$	$84.23_{\pm 0.20}$	$86.07_{\pm 0.20}$	$89.52_{\pm 0.31}$	81.09
	ENGINE	$87.00_{\pm 1.60}$	$75.82_{\pm 1.52}$	$90.08_{\pm 0.16}$	$74.69_{\pm 0.36}$	$85.44_{\pm 0.53}$	$68.87_{\pm 0.25}$	**$71.21_{\pm 0.77}$**	$84.09_{\pm 0.09}$	$86.98_{\pm 0.06}$	$89.05_{\pm 0.13}$	81.32
Reasoner	TAPE	$88.05_{\pm 1.76}$	$76.45_{\pm 1.60}$	$93.00_{\pm 0.13}$	$74.96_{\pm 0.14}$	$87.11_{\pm 0.66}$	$68.11_{\pm 0.54}$	$66.22_{\pm 0.83}$	$85.95_{\pm 0.59}$	**$87.72_{\pm 0.28}$**	**$90.46_{\pm 0.18}$**	**81.80**
Predictor	LLM_{IT}	$71.93_{\pm 1.47}$	$60.97_{\pm 3.97}$	$94.16_{\pm 0.19}$	**76.08**	$80.61_{\pm 0.47}$	$44.20_{\pm 3.06}$	$58.30_{\pm 0.48}$	$84.80_{\pm 0.13}$	$78.27_{\pm 0.54}$	$74.51_{\pm 0.53}$	72.38
	GraphGPT	$82.29_{\pm 0.26}$	$74.67_{\pm 1.15}$	$93.54_{\pm 0.22}$	$75.15_{\pm 0.14}$	$82.54_{\pm 0.23}$	$67.00_{\pm 1.22}$	$60.72_{\pm 1.47}$	$85.38_{\pm 0.72}$	$84.46_{\pm 0.36}$	$86.78_{\pm 1.14}$	79.25
	LLaGA	$87.55_{\pm 1.15}$	**$76.73_{\pm 1.70}$**	$90.28_{\pm 0.91}$	$74.49_{\pm 0.23}$	$84.03_{\pm 1.10}$	**$69.16_{\pm 0.72}$**	$71.06_{\pm 0.38}$	$85.56_{\pm 0.30}$	$87.62_{\pm 0.30}$	$90.41_{\pm 0.12}$	81.69

tive when labeled data is abundant. In supervised scenarios, LLM-as-Predictor methods enhance performance across most datasets. Especially, the LLaGA method achieves superior results among 5 of 10 datasets. Conversely, in semi-supervised settings, LLM-as-Predictor methods exhibit unstable performance, evidenced by low Macro-F1 scores, and imbalanced output distributions (detailed discussion in Appendix E.3). These findings indicate that LLM-as-Predictor methods are most effective when ample supervision is available, with LLaGA being an especially strong choice. Furthermore, within the predictor methods, LLM Instruction Tuning typically falls behind the other two methods and incurs substantial time costs (Table 17 and Table 18 in the Appendix). This shows that standalone LLMs are weak predictors and incorporating graph context is essential for achieving satisfactory performance.

4.2. Zero-shot Performance

Settings: For LLM Direct Inference, we utilize both closed-source and open-source LLMs, including GPT-4o (OpenAI, 2024), DeepSeek-Chat (Liu et al., 2024a), LLaMA3.1-8B (Dubey et al., 2024), and Mistral-7B. The prompt templates include **Direct**, **CoT**, **ToT**, and **ReAct** (Yao et al., 2023b). Additionally, we incorporate a node's neighboring information into extended prompts, referred to as "**w. Neighbor**", and have the LLMs first reason over neighbors to generate a summary that facilitates the subsequent classification task, referred to as "**w. Summary**". Prompt templates are listed in Appendix B.2. Besides, we assess the transferability of GFMs by evaluating **ZeroG** (Li et al., 2024b), **LLM Instruction Tuning**, and **LLaGA**. GFMs are applied following the intra-domain manner: each model is pre-trained on a larger dataset within the same domain (e.g., arXiv from the academic domain) before being evaluated on the target dataset. Results for Accuracy and Macro-F1 are shown in Table 2 and Table 11 in the Appendix, where we have the following takeaway:

Takeaway 6: GFMs can outperform open-source LLMs but still fall short of strong LLMs like GPT-4o. ZeroG outperforms LLaMA-8B in most cases, achieving up to a 6% average improvement in accuracy. However, it still falls short of GPT-4o and DeepSeek-Chat. Among GFMs, LLaGA performs poorly because it uses a projector to align the source graph's tokens with LLM input tokens. This projector may be dataset-specific, leading to reduced performance on different datasets, as also observed in Zhu et al. (2024a). These findings highlight the need for further research to improve the generalization of GFMs to match the performance of more powerful LLMs.

Takeaway 7: LLM direct inference can be improved by appropriately incorporating structural information. Our results reveal that advanced prompt templates such as CoT, ToT, and ReAct, offer only minor performance improvements. Specifically, models like LLaMA exhibit limited instruction-following abilities, often producing unexpected and over-length outputs when processing complex prompts

Table 2: **Performance comparison under zero-shot setting with Accuracy (%) and Macro-F1 (%) reported.** Numbers in brackets represent the dataset's homophily ratio (%). Results of other LLMs are shown in Table 11 in Appendix.

Type & LLM	Method	**Cora** (82.52)		**WikiCS** (68.67)		**Instagram** (63.35)		**Photo** (78.50)		**Avg.**	
		Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
LLM GPT-4o	Direct	68.08	69.25	68.59	63.21	44.53	42.77	63.99	61.09	61.30	59.08
	CoT	68.89	69.86	70.75	**66.23**	**47.87**	**47.57**	61.61	60.62	62.28	61.07
	ToT	68.29	69.13	70.78	65.69	44.16	42.68	60.84	59.16	61.02	59.16
	ReAct	68.21	69.28	69.45	66.03	44.49	43.16	63.63	60.82	61.44	59.82
	w. Neighbor	70.30	71.44	69.69	64.51	42.42	39.79	69.93	68.55	63.09	61.07
	w. Summary	**71.40**	**72.13**	**70.90**	65.42	45.02	44.62	**72.63**	**70.84**	**64.99**	**63.25**
LLM LLaMA-8B	Direct	62.64	63.02	56.77	53.04	37.58	29.70	41.23	44.26	49.56	47.50
	CoT	62.04	62.61	58.88	56.00	42.00	39.06	44.22	47.13	51.78	51.20
	ToT	34.06	33.30	40.35	41.15	**45.33**	**45.27**	31.31	34.00	37.76	38.43
	ReAct	36.55	38.04	22.40	25.76	44.67	44.42	27.03	28.96	32.66	34.30
	w. Neighbor	64.55	64.41	59.43	54.16	36.98	28.32	45.49	50.44	51.61	49.33
	w. Summary	**64.69**	**64.62**	**62.69**	**56.40**	37.59	30.91	**48.11**	**52.20**	**53.27**	**51.03**
GFM	ZeroG	**62.55**	**57.56**	**62.71**	**57.87**	**50.71**	**50.43**	46.27	**51.52**	**55.56**	**54.35**
	LLM_{IT}	52.58	51.89	60.83	53.59	41.58	26.26	**49.23**	44.85	51.06	44.15
	LLaGA	18.82	8.49	8.20	8.29	47.93	47.70	39.18	4.71	28.53	17.30

such as ReAct (Wu et al., 2025). This makes parsing classification results challenging and leads to suboptimal performance. The advanced prompts are generally designed for broad reasoning tasks and lack graph- or classification-specific knowledge, thereby limiting their benefits for the node classification task. In contrast, enriched prompts that incorporate structural information, i.e., "w. Neighbor" and "w. Summary", demonstrate performance enhancements across LLMs. The performance gains are particularly evident on homophilic datasets such as Cora and Photo (3%-10%), where neighboring nodes are likely to share the same labels as the central node. High homophily means that information from neighboring nodes provides crucial clues about a central node's label, thereby improving classification performance. Among these enriched prompts, "**w. Summary**" is especially effective as it not only provides structural context but also leverages the self-reflection abilities of LLMs to further utilize structural information.

4.3. Computational Cost Analysis

We evaluate the training and inference times of various methods in both semi-supervised and supervised settings. Detailed training times are provided in Tables 17 and 18 in the Appendix, while inference times are presented in Table 19. All measurements were conducted on a single NVIDIA H100-80G GPU to ensure consistency.

Based on the results, we can conclude that classic methods are highly efficient, with GNNs typically converging within seconds (e.g., 5.2 seconds for $\text{GCN}_{\text{ShallowEmb}}$ on Pubmed) and LMs fine-tuning completed within minutes. In contrast, LLM-as-Reasoner approaches are the most time-consuming (e.g., 5.9 hours for TAPE on Pubmed) because they require generating reasoning text for each node and subsequently processing this augmented text through both an LM and a GNN. This three-stage computational process significantly extends the overall computation time. LLM-as-Encoder methods are the most efficient among LLM-based approaches (e.g., 13.4 minutes for $\text{GCN}_{\text{LLMEmb}}$ on Pubmed), utilizing LLMs solely for feature encoding, which allows GNN training to remain efficient and complete within minutes. Although LLM-as-Predictor methods are more efficient than Reasoner approaches, they still require hours for effective model training. Among predictor methods, LLaGA is the most efficient (e.g., 25.6 minutes on Pubmed) as it encodes both the node's textual and structural information into embeddings instead of processing raw text.

During inference, a significant efficiency gap remains between LLM-based and classic methods. Classic methods can complete the entire inference process for thousands of cases within milliseconds, making them suitable for industrial deployments that demand real-time responses. In contrast, LLM-based methods are limited to processing one case within the same timeframe, highlighting the urgent need to improve their efficiency.

5. Fine-grained Analysis Within Each Category

In this section, we present empirical results within each category. For LLM-as-Encoder, we explore the conditions under which LLMs outperform traditional LMs. Additionally, we examine how key components (e.g., model type and size) influence the effectiveness of LLM-as-Reasoner and LLM-as-Predictor.

5.1. LLM-as-Encoder: Compared with LMs

Motivation and Settings: Both LLMs and small-scale LMs can encode nodes' associated texts. This raises the question:

Table 3: **Comparison of LLM- and LM-as-Encoder with Accuracy (%) reported under semi-supervised setting. LLM-as-Encoder outperforms LMs in heterophilic graphs.**

The **best encoder** within each method on a dataset is highlighted. Results in supervised settings are shown in Table 12 in Appendix.

Method	Encoder	Computer	Cora	Pubmed	Photo	Books	Citeseer	WikiCS	Instagram	Reddit
Homophily Ratio (%)		85.28	82.52	79.24	78.50	78.05	72.93	68.67	63.35	55.52
MLP	SenBERT	**69.57**$_{\pm 0.18}$	64.61$_{\pm 1.34}$	74.67$_{\pm 0.63}$	72.28$_{\pm 0.36}$	**81.93**$_{\pm 0.08}$	66.83$_{\pm 0.58}$	71.48$_{\pm 0.33}$	64.98$_{\pm 0.38}$	57.23$_{\pm 0.51}$
	RoBERTa	69.42$_{\pm 0.10}$	73.84$_{\pm 0.55}$	73.21$_{\pm 0.78}$	72.95$_{\pm 0.34}$	81.71$_{\pm 0.14}$	**70.59**$_{\pm 0.31}$	75.82$_{\pm 0.13}$	66.39$_{\pm 0.24}$	59.66$_{\pm 0.53}$
	Qwen-3B	67.54$_{\pm 0.29}$	**74.03**$_{\pm 0.57}$	75.30$_{\pm 0.72}$	72.72$_{\pm 0.23}$	81.60$_{\pm 0.53}$	68.26$_{\pm 0.79}$	78.64$_{\pm 0.37}$	66.53$_{\pm 0.37}$	60.49$_{\pm 0.17}$
	Mistral-7B	69.37$_{\pm 0.28}$	73.90$_{\pm 0.59}$	**75.70**$_{\pm 1.00}$	**74.16**$_{\pm 0.24}$	81.91$_{\pm 0.25}$	69.66$_{\pm 0.38}$	**79.56**$_{\pm 0.41}$	**66.68**$_{\pm 0.24}$	**61.91**$_{\pm 0.21}$
GCN	SenBERT	**88.92**$_{\pm 0.19}$	81.76$_{\pm 0.75}$	78.24$_{\pm 0.66}$	85.18$_{\pm 0.16}$	**83.47**$_{\pm 0.20}$	70.97$_{\pm 0.81}$	80.41$_{\pm 0.18}$	65.78$_{\pm 0.14}$	64.97$_{\pm 0.82}$
	RoBERTa	88.90$_{\pm 0.14}$	**84.56**$_{\pm 0.41}$	78.08$_{\pm 0.52}$	**85.19**$_{\pm 0.17}$	83.22$_{\pm 0.29}$	**73.52**$_{\pm 0.58}$	80.97$_{\pm 0.22}$	66.64$_{\pm 0.21}$	65.69$_{\pm 1.01}$
	Qwen-3B	87.55$_{\pm 0.14}$	83.62$_{\pm 0.41}$	78.50$_{\pm 0.80}$	84.26$_{\pm 0.33}$	82.83$_{\pm 0.24}$	71.50$_{\pm 0.92}$	**81.02**$_{\pm 0.33}$	66.69$_{\pm 0.59}$	**69.40**$_{\pm 0.56}$
	Mistral-7B	88.22$_{\pm 0.16}$	83.33$_{\pm 0.75}$	**78.71**$_{\pm 0.45}$	84.84$_{\pm 0.47}$	83.03$_{\pm 0.34}$	71.39$_{\pm 0.90}$	80.94$_{\pm 0.16}$	**67.49**$_{\pm 0.43}$	68.65$_{\pm 0.75}$
SAGE	SenBERT	**89.08**$_{\pm 0.06}$	80.45$_{\pm 0.79}$	77.29$_{\pm 0.45}$	85.54$_{\pm 0.16}$	**83.93**$_{\pm 0.17}$	69.42$_{\pm 1.42}$	80.02$_{\pm 0.24}$	65.34$_{\pm 0.44}$	61.65$_{\pm 0.17}$
	RoBERTa	88.97$_{\pm 0.09}$	**84.06**$_{\pm 0.52}$	75.82$_{\pm 0.59}$	**85.57**$_{\pm 0.17}$	83.74$_{\pm 0.22}$	**72.58**$_{\pm 0.45}$	80.77$_{\pm 0.29}$	66.53$_{\pm 0.50}$	63.65$_{\pm 0.32}$
	Qwen-3B	86.24$_{\pm 0.32}$	83.31$_{\pm 0.63}$	76.76$_{\pm 0.35}$	84.28$_{\pm 0.42}$	82.84$_{\pm 0.31}$	71.11$_{\pm 0.98}$	80.85$_{\pm 0.22}$	66.73$_{\pm 0.37}$	63.82$_{\pm 0.38}$
	Mistral-7B	88.48$_{\pm 0.20}$	82.73$_{\pm 0.99}$	**77.64**$_{\pm 1.73}$	85.50$_{\pm 0.20}$	83.32$_{\pm 0.16}$	71.42$_{\pm 0.47}$	**81.47**$_{\pm 0.32}$	**67.44**$_{\pm 0.06}$	**65.02**$_{\pm 0.13}$

Table 4: **Performance (%) of TAPE with different LLM backbones under semi-supervised setting**.

Metrics	LLM	Cora	Citeseer	Pubmed	WikiCS	Instagram	Reddit	Books	Photo	Computer	Avg.
Acc	Mistral	84.04$_{\pm 0.24}$	71.87$_{\pm 0.35}$	78.61 $_{\pm 1.23}$	**81.94**$_{\pm 0.16}$	66.07$_{\pm 0.10}$	**62.43**$_{\pm 0.47}$	84.92$_{\pm 0.26}$	86.46$_{\pm 0.12}$	89.52$_{\pm 0.04}$	78.43
	GPT-4o	**84.30**$_{\pm 0.36}$	**73.75**$_{\pm 0.67}$	**82.70**$_{\pm 1.78}$	81.93$_{\pm 0.33}$	**66.25**$_{\pm 0.38}$	62.22$_{\pm 1.24}$	**85.08**$_{\pm 0.17}$	**86.65**$_{\pm 0.17}$	**89.62**$_{\pm 0.13}$	**79.17**
F1	Mistral	81.89$_{\pm 0.31}$	66.80$_{\pm 0.33}$	78.46$_{\pm 1.13}$	80.03$_{\pm 0.23}$	50.01$_{\pm 1.60}$	**61.23**$_{\pm 0.69}$	47.12$_{\pm 3.26}$	82.31$_{\pm 0.19}$	**84.90**$_{\pm 1.14}$	70.31
	GPT-4o	**82.62**$_{\pm 0.60}$	**67.41**$_{\pm 0.82}$	**82.45**$_{\pm 1.65}$	**80.27**$_{\pm 0.34}$	**51.16**$_{\pm 3.54}$	61.11$_{\pm 1.52}$	**47.51**$_{\pm 2.92}$	**82.54**$_{\pm 0.18}$	84.28$_{\pm 2.98}$	**71.04**

Table 5: **Performance (%) of TAPE with different LLM backbones under supervised setting**.

Metrics	LLM	Cora	Citeseer	Pubmed	arXiv	WikiCS	Instagram	Reddit	Books	Photo	Computer	Avg.
Acc	Mistral	88.05$_{\pm 1.76}$	**76.45**$_{\pm 1.60}$	93.00$_{\pm 0.13}$	74.96$_{\pm 0.14}$	**87.11**$_{\pm 0.66}$	**68.11**$_{\pm 0.54}$	66.22$_{\pm 0.83}$	85.95$_{\pm 0.59}$	**87.72**$_{\pm 0.28}$	90.46$_{\pm 0.18}$	81.80
	GPT-4o	**88.24**$_{\pm 1.23}$	76.41$_{\pm 1.38}$	**94.12**$_{\pm 0.03}$	**75.08**$_{\pm 0.08}$	87.10$_{\pm 0.62}$	67.99$_{\pm 0.51}$	**66.33**$_{\pm 0.89}$	**86.19**$_{\pm 0.60}$	87.65$_{\pm 0.47}$	**90.56**$_{\pm 0.21}$	**81.97**
F1	Mistral	87.21$_{\pm 1.60}$	**73.33**$_{\pm 1.57}$	92.39$_{\pm 0.02}$	**57.79**$_{\pm 0.45}$	**86.03**$_{\pm 1.14}$	**58.31**$_{\pm 1.15}$	65.91$_{\pm 0.71}$	54.07$_{\pm 2.01}$	**83.41**$_{\pm 0.42}$	86.78$_{\pm 0.53}$	74.52
	GPT-4o	**87.34**$_{\pm 1.06}$	73.17$_{\pm 2.00}$	**93.58**$_{\pm 0.09}$	57.69$_{\pm 0.23}$	85.93$_{\pm 1.05}$	57.49$_{\pm 1.93}$	**66.09**$_{\pm 0.80}$	**54.32**$_{\pm 3.30}$	83.40$_{\pm 0.41}$	**86.91**$_{\pm 0.55}$	**74.59**

When do LLMs surpass LMs as encoders? To address this, we evaluate various methods using node features derived from LLMs and LMs, observing the resulting performance differences. For LMs, we select SenBERT-66M (Reimers & Gurevych, 2019) and RoBERTa-355M (Liu et al., 2019). For LLMs, we choose Qwen2.5-3B (Yang et al., 2024) and Mistral-7B (Jiang et al., 2023). The considered methods include: (1) **MLP**, which solely utilizes node features as input to predict labels without incorporating any graph information, (2) **GCN**, and (3) **GraphSAGE**. For each method, we input node features initialized from various LM or LLM backbones while keeping all other components consistent. From the results shown in Table 3 and Table 12 in the Appendix, we can observe that:

Takeaway 8: LLM-as-Encoder significantly outperforms LMs in heterophilic graphs. In heterophilic datasets such as Reddit, LLM-based encoders achieve 2% - 8% higher accuracy than their LM counterparts. This performance gap is most evident with the MLP method (4%–8%). The performance gain is less obvious in homophilic settings. We can also leverage the mutual information (1) to give theoretical insights. In homophilic graphs, edges often connect nodes with the same labels, whereas this does not hold in heterophilic graphs. Therefore, in homophilic graphs, the first term in (1) dominates, and the room for a better encoder, e.g., LLM, to improve is limited.

5.2. LLM-as-Reasoner: Impact of LLM Reasoning Capabilities

Motivation and Settings: In the LLM-as-Reasoner paradigm, as the language model should generate reasoning texts, the adopted language models should be autoregressive and the model size should be large. To investigate how the advanced reasoning capabilities of LLMs influence overall performance, we replace the default Mistral-7B model in the TAPE method with the more powerful GPT-4o model, keeping all other components unchanged.

Results and Analysis: As presented in Table 4 (semi-supervised settings) and Table 5 (supervised settings), the effectiveness of LLM-as-Reasoner methods positively correlates with the strength of the underlying LLMs. In semi-supervised settings, TAPE utilizing GPT-4o consistently

Table 6: **Accuracy (%) of LLaGA to different LLM backbones under supervised settings**.
The best LLM backbone within **each series** and **at similar scales** is highlighted. Semi-supervised performance is shown in Table 13.

	LLM	Cora	Citeseer	Pubmed	arXiv	WikiCS	Instagram	Reddit	Books	Photo	Computer	Avg.
Same series	Qwen-3B	84.91±2.19	74.83±2.46	88.61±1.24	71.82±1.37	82.23±3.14	62.49±0.98	67.96±0.90	83.56±1.86	**85.20±1.63**	**89.37±0.29**	79.10
	Qwen-7B	85.33±1.50	70.75±5.18	**90.53±0.49**	71.60±1.59	82.57±1.67	63.86±2.76	**68.62±0.53**	**84.23±0.51**	83.55±1.35	87.21±1.88	78.82
	Qwen-14B	**87.25±1.63**	**75.49±2.03**	89.93±0.27	**73.15±0.74**	82.26±1.51	63.88±2.49	67.60±1.77	83.94±0.41	84.83±0.77	87.06±0.80	79.54
	Qwen-32B	85.93±0.99	75.39±1.90	89.97±0.26	72.84±0.67	**83.49±0.91**	**64.33±1.69**	68.47±0.09	84.18±0.29	84.77±0.23	88.49±0.49	**79.79**
Similar scales	Mistral-7B	**87.55±1.15**	**76.73±1.70**	90.28±0.91	**74.49±0.23**	**84.03±1.10**	**69.16±0.72**	**71.06±0.38**	**85.56±0.30**	**87.62±0.30**	**90.41±0.12**	**81.69**
	Qwen-7B	85.33±1.50	70.75±5.18	**90.53±0.49**	70.47±1.12	82.57±1.67	63.86±2.76	68.62±0.53	84.23±0.51	83.55±1.35	87.21±1.88	78.71
	LLaMA-8B	85.77±1.34	74.84±1.09	89.57±0.24	72.72±0.26	82.25±1.65	61.12±0.45	67.70±0.44	84.05±0.26	85.57±0.41	89.42±0.12	79.30

Table 7: **Macro-F1(%) of LLaGA to different LLM backbones under supervised settings**.

	LLM	Cora	Citeseer	Pubmed	arXiv	WikiCS	Instagram	Reddit	Books	Photo	Computer	Avg.
Same-series	Qwen-3B	77.92±6.14	66.52±5.69	78.88±10.43	51.30±0.83	78.81±7.68	50.93±7.72	65.77±1.38	**49.87±1.52**	77.51±3.24	**80.77±3.27**	67.83
	Qwen-7B	82.50±4.12	64.03±4.86	**90.29±0.52**	51.97±0.83	77.35±4.26	56.50±1.15	**68.55±0.60**	46.21±1.78	75.76±5.34	78.86±5.90	69.20
	Qwen-14B	**85.64±1.89**	**69.92±3.95**	89.69±0.39	**53.32±0.38**	**79.13±1.78**	**57.58±0.83**	67.10±2.18	44.26±2.27	76.09±2.71	80.17±4.74	**70.29**
	Qwen-32B	82.85±4.10	68.11±3.69	89.57±0.37	52.52±0.65	77.31±3.89	57.28±2.21	68.22±0.02	48.25±1.29	**79.51±0.87**	77.15±7.53	70.08
Similar scales	Mistral-7B	**84.97±3.97**	**72.59±1.70**	90.00±0.80	**58.08±0.29**	**82.37±1.73**	**57.96±2.40**	62.14±15.59	**54.89±2.29**	**83.56±0.40**	**86.97±0.34**	**73.35**
	Qwen-7B	82.50±4.12	64.03±4.86	**90.29±0.52**	45.74±9.78	77.35±4.26	56.50±1.15	**68.55±0.60**	46.21±1.78	75.76±5.34	78.86±5.90	69.09
	LLaMA-8B	81.40±5.46	69.87±3.68	89.30±0.23	55.23±0.59	80.14±2.09	54.58±1.24	67.40±0.61	51.65±0.17	78.87±2.38	85.54±0.59	71.40

outperforms its Mistral-7B counterpart, achieving performance gains of up to 4% on the Pubmed dataset. However, in supervised scenarios, the performance gap between GPT-4o and Mistral-7B narrows. This reduction is attributed to the abundance of labeled data, which increases the mutual information $I(\mathcal{E}; \mathcal{Y}_l)$ in (1). Consequently, the dependency on node attributes decreases, thereby diminishing the relative advantages of more powerful LLMs.

Based on these findings, we recommend that when abundant supervision is available, practitioners may opt for open-source LLMs instead of more powerful and costlier models for the LLM-as-Reasoner method. This practice can achieve comparable performance without incurring additional costs.

5.3. LLM-as-Predictor: Sensitivity to LLM Backbones

Motivation and Settings: For most LLM-as-Predictor methods, only open-source LLMs are compatible. Given the diverse choices and varying scales of these models, we aim to investigate the sensitivity of performance to different LLM backbones. This examination seeks to identify potential scaling laws and determine which LLMs excel at the node classification task. Therefore, we choose the best predictor method LLaGA as the baseline, include models of different sizes within the same series, i.e., Qwen2.5-series (Yang et al., 2024). Additionally, we consider similar-scaled models to identify the most suitable for this task, including Qwen2.5-7B, Mistral-7B, and LLaMA3.1-8B. All experiments maintain consistency by only varying the backbone LLMs while keeping other components, training configurations, and hyperparameters unchanged.

Results and Analysis: (1) Scaling within the same series: Comparing Qwen-3B to Qwen-32B (performance shown in Tables 6 and 7, efficiency in Table 14, and performance trends in Figures 3 and 4 in the Appendix), we observe that performance generally improves with larger model sizes. However, beyond Qwen-7B and Qwen-14B, the performance gains become marginal while training and inference times increase significantly. For instance, Qwen-32B takes over 200 milliseconds per sample for inference, which is five times longer than Qwen-7B. Therefore, Qwen-7B or Qwen-14B are recommended as practical choices balancing performance and efficiency. **(2) Model selection at similar scales:** When comparing models of similar sizes (Tables 6 and 7, and Table 13 in the Appendix), Mistral-7B outperforms other LLMs of comparable scale. Its superior performance makes Mistral-7B the recommended backbone LLM for node classification tasks.

6. Conclusion

This paper provides guidelines for leveraging LLMs to enhance node classification tasks across diverse real-world applications. We introduce LLMNodeBed, a codebase and testbed for systematic comparisons, featuring ten datasets, eight LLM-based algorithms, and three learning paradigms. Through extensive experiments involving 2,200 models, we uncover key insights: In supervised settings, each category offers unique advantages, but LLM-based approaches deliver marginal improvements over classic methods when ample supervision is available. In zero-shot scenarios, directing powerful LLMs to perform inference with integrated structural context yields the best performance.

Our findings offer practical guidance for practitioners applying LLMs to node classification tasks and highlight research gaps, e.g., the limited exploration of LLMs on heterophilic graphs and the scarcity of such text-rich datasets. We intend to address these gaps in future work. We hope that LLMNodeBed will inspire and serve as a valuable toolkit for further research.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(11), 2006.

Chai, Z., Zhang, T., Wu, L., Han, K., Hu, X., Huang, X., and Yang, Y. GraphLLM: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*, 2023.

Chen, R., Zhao, T., Jaiswal, A. K., Shah, N., and Wang, Z. LLaGA: Large language and graph assistant. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 7809–7823, 2024a.

Chen, Z., Mao, H., Li, H., Jin, W., Wen, H., Wei, X., Wang, S., Yin, D., Fan, W., Liu, H., et al. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2): 42–61, 2024b.

Chen, Z., Mao, H., Wen, H., Han, H., Jin, W., Zhang, H., Liu, H., and Tang, J. Label-free node classification on graphs with large language models (llms), 2024c. URL https://arxiv.org/abs/2310.04668.

Dai, X., Qu, H., Shen, Y., Zhang, B., Wen, Q., Fan, W., Li, D., Tang, J., and Shan, C. How do large language models understand graph patterns? a benchmark for graph pattern comprehension, 2024. URL https://arxiv.org/abs/2410.05298.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., and et al, A. F. The llama 3 herd of models. *ArXiv*, abs/2407.21783, 2024.

Giles, C. L., Bollacker, K. D., and Lawrence, S. Citeseer: an automatic citation indexing system. In *Digital library*, 1998.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *NIPS*, 2017.

He, X., Bresson, X., Laurent, T., Perold, A., LeCun, Y., and Hooi, B. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning, 2023.

Hu, J. E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., and Chen, W. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Huang, J., Zhang, X., Mei, Q., and Ma, J. Can llms effectively leverage graph structural information through prompts, and why? *Trans. Mach. Learn. Res.*, 2024, 2023. URL https://api.semanticscholar.org/CorpusID:268032402.

Huang, X., Han, K., Yang, Y., Bao, D., Tao, Q., Chai, Z., and Zhu, Q. Can gnn be good adapter for llms? *Proceedings of the ACM on Web Conference 2024*, 2024.

Ji, Y., Liu, C., Chen, X., Ding, Y., Luo, D., Li, M., Lin, W., and Lu, H. Nt-llm: A novel node tokenizer for integrating graph structure into large language models. *ArXiv*, abs/2410.10743, 2024.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de Las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b. *ArXiv*, abs/2310.06825, 2023.

Jin, B., Liu, G., Han, C., Jiang, M., Ji, H., and Han, J. Large language models on graphs: A comprehensive survey. *arXiv preprint arXiv:2312.02783*, 2023a.

Jin, B., Zhang, W., Zhang, Y., Meng, Y., Zhang, X., Zhu, Q., and Han, J. Patton: Language model pretraining on text-rich networks. In *61st Annual Meeting of the Association for Computational Linguistics, ACL 2023*, pp. 7005–7020. Association for Computational Linguistics (ACL), 2023b.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Kong, L., Feng, J., Liu, H., Huang, C., Huang, J., Chen, Y., and Zhang, M. Gofa: A generative one-for-all model for joint graph language modeling. *ArXiv*, abs/2407.09709, 2024.

Li, X., Chen, W., Chu, Q., Li, H., Sun, Z., Li, R., Qian, C., Wei, Y., Liu, Z., Shi, C., Sun, M., and Yang, C. Can large language models analyze graphs like professionals? a benchmark, datasets and models, 2024a. URL https://arxiv.org/abs/2409.19667.

Li, Y., Li, Z., Wang, P., Li, J., Sun, X., Cheng, H., and Yu, J. X. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*, 2023.

Li, Y., Wang, P., Li, Z., Yu, J. X., and Li, J. Zerog: Investigating cross-dataset zero-shot transferability in graphs. *arXiv preprint arXiv:2402.11235*, 2024b.

Li, Y., Wang, P., Zhu, X., Chen, A., Jiang, H., Cai, D., Chan, V. W. K., and Li, J. Glbench: A comprehensive benchmark for graph with large language models. In *Proceedings of Neural Information Processing Systems*, 2024c.

Liu, A., Feng, B., Wang, B., Wang, B., Liu, B., Zhao, C., Dengr, C., Ruan, C., Dai, D., and et al., D. G. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024a. URL https://arxiv.org/abs/2405.04434.

Liu, H., Feng, J., Kong, L., Liang, N., Tao, D., Chen, Y., and Zhang, M. One for all: Towards training one graph model for all classification tasks. In *The Twelfth International Conference on Learning Representations*, 2024b.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

Luo, L., Li, Y.-F., Haffari, G., and Pan, S. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, 2024a.

Luo, Y., Shi, L., and Wu, X.-M. Classic GNNs are strong baselines: Reassessing GNNs for node classification. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024b. URL https://openreview.net/forum?id=xkljKdGe4E.

Ma, Y. and Tang, J. *Deep Learning on Graphs*. Cambridge University Press, 2021.

Mao, K., Xiao, X., Zhu, J., Lu, B., Tang, R., and He, X. Item tagging for information retrieval: A tripartite graph neural network based approach. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

Mernyei, P. and Cangea, C. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *ArXiv*, abs/2007.02901, 2020.

Ni, J., Li, J., and McAuley, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Conference on Empirical Methods in Natural Language Processing*, 2019.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Perozzi, B., Fatemi, B., Zelle, D., Tsitsulin, A., Kazemi, M., Al-Rfou, R., and Halcrow, J. Let your graph do the talking: Encoding structured data for llms, 2024.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 11 2019.

Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.*, 24:513–523, 1988.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., and Eliassi-Rad, T. Collective classification in network data. In *The AI Magazine*, 2008.

Shi, Y., Huang, Z., Wang, W., Zhong, H., Feng, S., and Sun, Y. Masked label prediction: Unified massage passing model for semi-supervised classification. *ArXiv*, abs/2009.03509, 2020.

Tang, J., Yang, Y., Wei, W., Shi, L., Su, L., Cheng, S., Yin, D., and Huang, C. GraphGPT: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023*, 2023.

Tang, J., Zhang, Q., Li, Y., and Li, J. Grapharena: Benchmarking large language models on graph computational problems, 2024. URL https://arxiv.org/abs/2407.00379.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. *International Conference on Learning Representations*, 2018.

Wang, H., Feng, S., He, T., Tan, Z., Han, X., and Tsvetkov, Y. Can language models solve graph problems in natural language? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=UDqHhbqYJV.

Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. Text embeddings by weakly-supervised contrastive pre-training. *ArXiv*, abs/2212.03533, 2022.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E. H., Xia, F., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. In *Proceedings of Neural Information Processing Systems*, 2022.

Wen, Z. and Fang, Y. Augmenting low-resource text classification with graph-grounded pre-training and prompting. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023.

Wu, F., Zhang, T., de Souza, A. H., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, 2019.

Wu, J., Yin, W., Jiang, Y., Wang, Z., Xi, Z., Fang, R., Zhou, D., Xie, P., and Huang, F. Webwalker: Benchmarking llms in web traversal, 2025. URL https://arxiv.org/abs/2501.07572.

Wu, X., Shen, Y., Shan, C., Song, K., Wang, S., Zhang, B., Feng, J., Cheng, H., Chen, W., Xiong, Y., and Li, D. Can graph learning improve planning in llm-based agents? In *Proceedings of Neural Information Processing Systems*, 2024.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Yan, H., Li, C., Long, R., Yan, C., Zhao, J., Zhuang, W., Yin, J., Zhang, P., Han, W., Sun, H., et al. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264, 2023.

Yan, Q., Zhang, Y., Liu, Q., Wu, S., and Wang, L. Relation-aware heterogeneous graph for user profiling. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., Xu, J., Zhou, J., Bai, J., He, J., Lin, J., Dang, K., Lu, K., Chen, K.-Y., Yang, K., Li, M., Xue, M., Ni, N., Zhang, P., Wang, P., Peng, R., Men, R., Gao, R., Lin, R., Wang, S., Bai, S., Tan, S., Zhu, T., Li, T., Liu, T., Ge, W., Deng, X., Zhou, X., Ren, X., Zhang, X., Wei, X., Ren, X., Fan, Y., Yao, Y., Zhang, Y., Wan, Y., Chu, Y., Cui, Z., Zhang, Z., and Fan, Z.-W. Qwen2 technical report. *ArXiv*, abs/2407.10671, 2024.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016a.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. *ArXiv*, abs/1603.08861, 2016b.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. Tree of Thoughts: Deliberate problem solving with large language models, 2023a.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b.

Zhang, G., Li, Z., Huang, J., Wu, J., Zhou, C., Yang, J., and Gao, J. efraudcom: An e-commerce fraud detection system via competitive graph neural networks. *ACM Transactions on Information Systems (TOIS)*, 40:1 – 29, 2022a.

Zhang, S., Liu, Y., Sun, Y., and Shah, N. Graph-less neural networks: Teaching old mlps new tricks via distillation. In *International Conference on Learning Representations*, 2022b. URL https://arxiv.org/abs/2110.08727.

Zhang, Z., Wang, X., Zhang, Z., Li, H., Qin, Y., and Zhu, W. Llm4dyg: Can large language models solve spatial-temporal problems on dynamic graphs? In *Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, 2024.

Zhao, J., Qu, M., Li, C., Yan, H., Liu, Q., Li, R., Xie, X., and Tang, J. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations*, 2023a.

Zhao, J., Zhuo, L., Shen, Y., Qu, M., Liu, K., Bronstein, M. M., Zhu, Z., and Tang, J. Graphtext: Graph reasoning in text space. *ArXiv*, abs/2310.01089, 2023b.

Zhu, X., Ghahramani, Z., and Lafferty, J. D. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

Zhu, Y., Shi, H., Wang, X., Liu, Y., Wang, Y., Peng, B., Hong, C., and Tang, S. Graphclip: Enhancing transferability in graph foundation models for text-attributed graphs. *ArXiv*, abs/2410.10329, 2024a.

Zhu, Y., Wang, Y., Shi, H., and Tang, S. Efficient tuning and inference for large language models on textual graphs. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCA)*, 2024b.

A. Related Works and Discussion

In this section, we present a comprehensive taxonomy of node classification methods, ranging from classic approaches to those leveraging LLMs.

A.1. Classic Methods

Early approaches for node classification tasks relied on structural techniques such as Laplacian regularization (Belkin et al., 2006), graph embeddings (Yang et al., 2016a), and label propagation (Zhu et al., 2003). These methods infer node labels by leveraging the connectivity and similarity among nodes within the graph.

Over the past decade, GNNs have emerged as the dominant paradigm for node classification, demonstrating superior performance across various benchmarks (Kipf & Welling, 2017; Veličković et al., 2018; Hamilton et al., 2017; Xu et al., 2019). GNNs enhance node representations by aggregating and transforming feature information from their local neighborhoods. Formally, given a graph's feature matrix $\boldsymbol{X}$ and structure $\mathcal{E}$, a GNN produces the predicted label matrix as $\boldsymbol{Y} = \text{GNN}_\Theta(\boldsymbol{X}, \mathcal{E}) \in \mathbb{R}^{|\mathcal{V}| \times C}$, where C is the number of classes, and Θ represents the learned parameters.

Beyond structural methods, node classification can also be approached using LMs by treating each node as a text entity. Fine-tuning LMs (Liu et al., 2019; Reimers & Gurevych, 2019; Wang et al., 2022) allows these models to map textual information directly to node labels, leveraging their strong language understanding capabilities to predict labels. To harness the complementary strengths of GNNs and LMs, i.e., structural and textual information, hybrid LM+GNN architectures have been developed (Jin et al., 2023b; Zhao et al., 2023a; Wen & Fang, 2023). These models integrate LM-encoded textual features with GNN-processed structural features, enhancing node classification performance by combining both modalities.

A.2. LLM-based Methods

LLM as Encoder: LLMs possess an extensive number of parameters, enabling them to generate highly expressive representations. These representations can replace shallow node embeddings, promising to enhance expressiveness and improve the performance of downstream task. A notable approach is ENGINE (Zhu et al., 2024b), which utilizes hidden embeddings from LLMs to construct node embeddings. ENGINE integrates these embeddings with GNNs to propagate and update representations. Specifically, it aggregates hidden embeddings from each LLM layer that processes a node's text and incorporates them into a cascaded GNN structure.

LLM as Reasoner: A significant advantage of LLMs is their reasoning and planning capabilities (Luo et al., 2024a; Wu et al., 2024). Guided by carefully crafted prompts, LLMs can intelligently execute a variety of downstream tasks. Motivated by this strength, He et al. (2023) introduced TAPE, a method that leverages LLM as Reasoner for node classification. Specifically, TAPE prompts the LLM to generate predictions along with a chain-of-thought reasoning process that includes explanations, denoted as $s_v^{\text{exp}} = \text{LLM}(s_v^{\text{orig}}, p)$, where p denotes the textual prompt, and s_v^{orig} and s_v^{exp} represent the original and generated texts for node v, respectively. Both the original and generated texts are processed by an LM to produce embeddings as $\boldsymbol{x}_v^{\text{orig}} = \text{LM}(s_v^{\text{orig}})$ and $\boldsymbol{x}_v^{\text{exp}} = \text{LM}(s_v^{\text{exp}})$, which are subsequently processed by GNNs for the classification task as:

$$\boldsymbol{Y} = \text{Ensemble}(\text{GNN}_{\Theta_1}(\boldsymbol{X}^{\text{orig}}, \mathcal{E}), \text{GNN}_{\Theta_2}(\boldsymbol{X}^{\text{exp}}, \mathcal{E})).$$

Subsequent works have enhanced the reliability of LLM reasoning. For example, KEA (Chen et al., 2024b) prompts LLMs to extract and explain specific technical terms from a node's original text instead of making direct predictions, thereby mitigating potential misguidance. Overall, the LLM-as-Reasoner paradigm leverages LLMs' reasoning capabilities to generate reliable explanations, thus augmenting the original graph data.

LLM as Predictor: LLMs' strong reasoning abilities make them effective for direct downstream classification tasks. In the LLM-as-Predictor paradigm, a node's textual and structural information, along with task-specific instructions, are tokenized and input into an LLM for prediction. A notable method in this category is LLaGA (Chen et al., 2024a). Firstly, the original text s_v of node v is encoded via LM as $\boldsymbol{x}_v^{\text{LM}} = \text{LM}(s_v)$. Then, a parameter-free GNN, i.e., SGC (Wu et al., 2019), updates the node embeddings based on the graph structure, initializing with $\boldsymbol{h}_v^{(0)} = \boldsymbol{x}_v^{\text{LM}}$. The embeddings from each SGC layer are concatenated into $\boldsymbol{H}_v = [\boldsymbol{h}_v^{(0)}, \ldots, \boldsymbol{h}_v^{(L)}]$ and further projected into the LLM's dimensionality using a projection layer ϕ_θ. These projected embeddings are then combined with tokenized instructions $\boldsymbol{T}$ and input into the LLM to generate the

predicted label as:

$$\ell_v = \text{LLM}([\phi_\theta(\boldsymbol{H}_v) \parallel \boldsymbol{T}]).$$

In the LLaGA framework, only the parameters of the projection layer are tuned, utilizing the next-token-prediction loss based on ground-truth labels and generated outputs. GraphGPT employs a more complex framework with three distinct pre-training and instruction tuning stages. Other LLM-as-Predictor methods (Chai et al., 2023; Perozzi et al., 2024; Kong et al., 2024; Huang et al., 2024; Zhao et al., 2023b; Ji et al., 2024) share similar frameworks with LLaGA but vary in integration approaches, training objectives, and tackled tasks.

A.3. Zero-shot Learning with LLMs

Supervised learning approaches, which rely on labeled data, often struggle to keep pace with the rapid evolution of real-world graph data. Zero-shot learning methods address this limitation by enabling models to generalize to unseen data without requiring explicit labels. These methods can be broadly categorized into two approaches: LLM Direct Inference and GFMs.

LLM Direct Inference involves using LLMs to make predictions directly on the node's information through various prompt engineering techniques. Advanced prompt templates for reasoning tasks include Chain-of-Thought (Wei et al., 2022), ReAct (Yao et al., 2023b), and Tree-of-Thought (Yao et al., 2023a). Besides, structural information can also be integrated into extended prompts (Tang et al., 2023; Wang et al., 2023; Huang et al., 2023), enriching the input provided to LLMs and facilitating more accurate predictions.

On the other hand, **GFMs** are foundation models pre-trained on extensive graph corpora to achieve general graph intelligence. Approaches such as ZeroG (Li et al., 2024b) and OFA (Liu et al., 2024b) fine-tune LMs or GNNs on multiple graphs, enabling these models to generalize to unseen graph datasets without extensive retraining. There also exist other zero-shot learning methods utilizing LLMs. For example, Chen et al. (2024c) leverage LLMs as annotators to generate pseudo-labels for GNN training, enabling classification tasks. These approaches are not considered in this work, as our focus is primarily on LLMs directly solving node classification tasks in zero-shot scenarios.

A.4. Benchmarks of LLMs for Graphs

In addition to developments in node classification algorithms, we discuss existing benchmarks that leverage LLMs for graph-related tasks. These benchmarks can be categorized based on the type of tasks they address.

The first category primarily utilizes LLMs for basic graph reasoning tasks, e.g., shortest path and connectivity. For instance, NLGraph (Wang et al., 2023) is a pioneering benchmark that encompasses eight different graph reasoning tasks presented in natural language. LLM4DyG (Zhang et al., 2024) further extends these reasoning tasks to dynamic graph settings. GraphArena (Tang et al., 2024) deals with more complex graph computational problems, with the complexity of tasks ranging from polynomial to NP-Complete like the Traveling Salesman Problem. ProGraph (Li et al., 2024a) evaluates the scalability of LLMs by handling large graphs with up to 10^6 nodes, necessitating the use of Python APIs for graph analysis rather than relying solely on direct reasoning of LLMs. Additionally, Dai et al. (2024) investigates whether LLMs can recognize graph patterns, e.g., triangles or squares, based on terminological or topological descriptions.

The second category focuses on the potential of LLMs for node classification tasks. While numerous surveys discuss the progress in this area (Li et al., 2023; Jin et al., 2023a), benchmarks that systematically evaluate LLM-based node classification methods remain limited. In the preliminary work by Chen et al. (2024b), the exploration is confined to a narrow range of LLM-as-Encoder and LLM-as-Reasoner approaches, primarily focusing on a limited set of language models. GLBench (Li et al., 2024c) emerges as the first comprehensive benchmark for LLM-based node classification, offering consistent data splits to evaluate representative methods in both semi-supervised and zero-shot settings. However, variations in backbone models and implemented codebases impede fair and rigorous comparisons.

Our benchmark, LLMNodeBed, distinguishes itself from existing benchmarks like GLBench by standardizing implementations of baselines, extending learning paradigms and datasets to encompass more real-world contexts, and incorporating influential factors like model type and size, homophily, and prompt design. This comprehensive approach provides more practical guidelines for effectively leveraging LLMs to enhance node classification tasks.

B. Prompts

B.1. Prompt in LLM-as-Predictor Methods

For GraphGPT (Tang et al., 2023) and LLaGA (Chen et al., 2024a), we utilize the prompt templates provided in their original papers for several datasets, including Cora and arXiv. For datasets not originally addressed, such as Photo, we adapt their prompt designs to create similarly formatted prompts. For LLM Instruction Tuning, we carefully craft prompt templates tailored to each dataset to directly guide the LLMs in performing classification tasks. Below is a summary of these prompt templates using Cora as an example: ⟨**labels**⟩ denotes the dataset-specific label space (Table 9), ⟨**graph**⟩ represents the tokenized graph context, and ⟨**raw_text**⟩ refers to the node's original raw text.

Illustration of Prompts Utilized by LLM-as-Predictor Methods on the Cora Dataset

LLM Instruction Tuning: Given a node-centered graph with centric node description: ⟨**raw_text**⟩, each node represents a paper, we need to classify the center node into 7 classes: ⟨**labels**⟩, please tell me which class the center node belongs to?

GraphGPT: Given a citation graph: ⟨**graph**⟩, where the 0-th node is the target paper, with the following information: ⟨**raw_text**⟩. Question: Which of the following specific research does this paper belong to: ⟨**labels**⟩. Directly give the full name of the most likely category of this paper.

LLaGA: Given a node-centered graph: ⟨**graph**⟩, each node represents a paper, we need to classify the center node into 7 classes: ⟨**labels**⟩, please tell me which class the center node belongs to?

B.2. Prompt in Zero-shot Scenarios

For LLM Direct Inference, we consider prompt templates including Direct, Chain-of-Thought (Wei et al., 2022), Tree-of-Thought (Yao et al., 2023a), and ReACT (Yao et al., 2023b). The latter three methods are effective in enhancing LLM reasoning abilities across different tasks. Therefore, we adopt these advanced prompts for node classification to assess their continued effectiveness. Illustrations of these prompts on the Cora dataset are shown below:

Illustration of Advanced Prompts Utilized by LLM Inference on the Cora Dataset

Direct: Given the information of the node: ⟨**raw_text**⟩. Question: Which of the following categories does this paper belong to? Here are the categories: ⟨**labels**⟩. Reply only with one category that you think this paper might belong to. Only reply with the category name without any other words.

Chain-of-Thought: Given the information of the node: ⟨**raw_text**⟩. Question: Which of the following types does this paper belong to? Here are the 7 categories: ⟨**labels**⟩. **Let's think about it step by step.** Analyze the content of the node and choose one appropriate category. Output format: ⟨reason: ⟩, ⟨classification: ⟩

Tree-of-Thought: Given the information of the node: ⟨**raw_text**⟩. Imagine three different experts answering this question. All experts will write down 1 step of their thinking, and then share it with the group. Then all experts will go on to the next step, etc. If any expert realizes they're wrong at any point then they leave. Question: Based on this information, which of the following categories does this paper belong to? Here are 7 categories: ⟨**labels**⟩. **Let's think through this using a tree of thought approach.** Output format: ⟨discussion: ⟩, ⟨classification: ⟩. The classification should only consist of one of the category names listed.

ReACT: Given the information of the node: ⟨**raw_text**⟩. Your task is to determine which of the following categories this paper belongs to. Here are the 7 categories: ⟨**labels**⟩. **Solve this question by interleaving the Thought, Action, and Observation steps.** Thought can reason about the current situation, and Action can be one of the following: (1) Search[entity], which searches the exact entity on Wikipedia and returns the first paragraph if it exists. If not, it will return some similar entities to search. (2) Lookup[keyword], which returns the next sentence containing the keyword in the current passage. (3) Finish[answer], which returns the answer and finishes the task. The output format must be ⟨process: ⟩, ⟨classification: ⟩. The classification should only consist of one of the category names listed.

Additionally, we incorporate the central node's structural information into extended prompts to evaluate performance. We propose two variants for integrating neighbor information: (1) "w. Neighbor": we concatenate the texts of all 1-hop neighbors of the central node as enriched context, and (2) "w. Summary": we first provide all neighbors' information to LLMs to generate a summary that highlights the common points among the neighbors. Then, we feed both the generated summary and the node's text to LLMs to facilitate prediction. The prompt templates for these methods on the Cora dataset are illustrated below:

Illustration of Structure-enriched Prompts Utilized by LLM Inference on the Cora Dataset

w. Neighbor: Given the information of the node: ⟨**raw_text**⟩. Given the information of its neighbors ⟨**raw_text**⟩. Here I give you the content of the node itself and the information of its 1-hop neighbors. The relation between the node and its neighbors is 'citation'. Question: Based on this information, Which of the following sub-categories of AI does this paper belong to? Here are the 7 categories: ⟨**labels**⟩. Reply only one category that you think this paper might belong to. Only reply with the category name without any other words.

w. Summary (Step 1): The following list records papers related to the current one, with the relationship being 'citation': ⟨**raw_text**⟩. Please summarize the information above with a short paragraph, and find some common points that can reflect the category of the paper.

w. Summary (Step 2): Given the information of the node: ⟨**raw_text**⟩, ⟨**summary**⟩. Here I give you the content of the node itself and the summary information of its 1-hop neighbors. The relation between the node and its neighbors is 'citation'. Question: Based on this information, Which of the following sub-categories of AI does this paper belong to? Here are the 7 categories: ⟨**labels**⟩. Reply only one category that you think this paper might belong to. Only reply with the category name without any other words.

C. Supplementary Materials for LLMNodeBed

C.1. Datasets

Table 8: **Statistics of supported datasets in LLMNodeBed.**

Statistics	Academic				Web Link	Social		E-Commerce		
	Cora	Citeseer	Pubmed	arXiv	WikiCS	Instagram	Reddit	Books	Photo	Computer
# Classes	7	6	3	40	10	2	2	12	12	10
# Nodes	2,708	3,186	19,717	169,343	11,701	11,339	33,434	41,551	48,362	87,229
# Edges	5,429	4,277	44,338	1,166,243	215,863	144,010	198,448	358,574	500,928	721,081
Avg. # Token	183.4	210.0	446.5	239.8	629.9	56.2	197.3	337.0	201.5	123.1
Homophily (%)	82.52	72.93	79.24	63.53	68.67	63.35	55.52	78.05	78.50	85.28

We selected 10 datasets from academic, web link, social, and e-commerce domains to create a diverse graph database. Within LLMNodeBed, each dataset is stored in `.pt` format using PyTorch, which includes shallow embeddings, raw text of nodes, edge indices, labels, and data splits for convenient loading. The datasets are described below, while their statistics and additional details are provided in Table 8 and Table 9, respectively.

- **Academic Networks:** The **Cora** (Sen et al., 2008), **Citeseer** (Giles et al., 1998), **Pubmed** (Yang et al., 2016b), and **ogbn-arXiv** (abbreviated as "arXiv") (Hu et al., 2020) datasets consist of nodes representing papers, with edges indicating citation relationships. The associated text attributes include each paper's title and abstract, which we use the collected version as follows: Cora and Pubmed from He et al. (2023), Citeseer from Chen et al. (2024b). Within the dataset, each node is labeled according to its category. For example, the arXiv dataset includes 40 CS sub-categories such as cs.AI (Artificial Intelligence) and cs.DB (Databases).
- **Web Link Network:** In the **WikiCS** dataset (Mernyei & Cangea, 2020), each node represents a Wikipedia page, and edges indicate reference links between pages. The raw text for each node includes the page name and content, which was collected by Liu et al. (2024b). The classification goal is to categorize each entity into different Wikipedia categories.
- **Social Networks:** The **Reddit** and **Instagram** datasets, originally released in Huang et al. (2024), feature nodes representing users, with edges denoting social connections like following relationships. For Reddit, each user's associated text consists of their historically published sub-reddits, while for Instagram, it includes the user's profile page introduction. In Reddit, nodes are labeled to indicate whether the user is popular or normal, while in Instagram, labels specify whether a user is commercial or normal.
- **E-Commerce Networks:** The **Ele-Photo** (abbreviated as "Photo") and **Ele-Computer** (abbreviated as "Computer") datasets are derived from the Amazon Electronics dataset (Ni et al., 2019), where each node represents an item in the Photo or Computer category. The **Books-History** (abbreviated as "Books") dataset comes from the Amazon Books dataset, where each node corresponds to a book in the history category. We utilize the processed datasets released in Yan et al. (2023). In these e-commerce networks, edges indicate co-purchase or co-view relationships. The associated text for each item includes descriptions, e.g., book titles and summaries, or user reviews. The classification task involves categorizing these products into fine-grained sub-categories.

C.2. Implementation Details and Hyperparameters Setting

- For **GNNs** with arbitrary input embeddings, either from shallow embeddings or those generated by LMs or LLMs, we perform a grid-search on the hyperparameters as follows:

  `num_layers` in $[2, 3, 4]$, `hidden_dimension` in $[32, 64, 128, 256]$, and `dropout` in $[0.3, 0.5, 0.7]$.

  Additionally, we explore the design space by considering the inclusion or exclusion of `batch_normalization` and `residual_connection`.

  For **shallow embeddings**, the Cora, Citeseer, Pubmed, WikiCS, and arXiv datasets provide initialized embeddings in their released versions (Hu et al., 2020; Sen et al., 2008). For remaining datasets lacking shallow embeddings, we construct these embeddings using **Node2Vec** (Grover & Leskovec, 2016) techniques, generating a fixed 300-dimensional embedding for each node based on a walk length of 30 and a total of 10 walks.

Table 9: **Details of datasets: label space and training data percentages with supervision**.

Domain	Dataset	Label Space
Academic	Cora	Rule_Learning, Neural_Networks, Case_Based, Genetic_Algorithms, Theory, Reinforcement_Learning, Probabilistic_Methods
	Citeseer	Agents, ML (Machine Learning), IR (Information Retrieval), DB (Databases), HCI (Human-Computer Interaction), AI (Artificial Intelligence)
	Pubmed	Experimentally induced diabetes, Type 1 diabetes, Type 2 diabetes
	arXiv	cs.NA, cs.MM, cs.LO, cs.CY, cs.CR, cs.DC, cs.HC, cs.CE, cs.NI, cs.CC, cs.AI, cs.MA, cs.GL, cs.NE, cs.SC, cs.AR, cs.CV, cs.GR, cs.ET, cs.SY, cs.CG, cs.OH, cs.PL, cs.SE, cs.LG, cs.SD, cs.SI, cs.RO, cs.IT, cs.PF, cs.CL, cs.IR, cs.MS, cs.FL, cs.DS, cs.OS, cs.GT, cs.DB, cs.DL, cs.DM
Web Link	WikiCS	Computational Linguistics, Databases, Operating Systems, Computer Architecture, Computer Security, Internet Protocols, Computer File Systems, Distributed Computing Architecture, Web Technology, Programming Language Topics
Social	Instagram	Normal User, Commercial User
	Reddit	Normal User, Popular User
E-Commerce	Books	World, Americas, Asia, Military, Europe, Russia, Africa, Ancient Civilizations, Middle East, Historical Study & Educational Resources, Australia & Oceania, Arctic & Antarctica
	Photo	Video Surveillance, Accessories, Binoculars & Scopes, Video, Lighting & Studio, Bags & Cases, Tripods & Monopods, Flashes, Digital Cameras, Film Photography, Lenses, Underwater Photography
	Computer	Computer Accessories & Peripherals, Tablet Accessories, Laptop Accessories, Computers & Tablets, Computer Components, Data Storage, Networking Products, Monitors, Servers, Tablet Replacement Parts

Setting	Cora	Citeseer	Pubmed	arXiv	WikiCS	Instagram	Reddit	Books	Photo	Computer
Semi-supervised	5.17%	3.77%	0.30%	-	4.96%	10.00%	10.00%	10.00%	10.00%	10.00%
Supervised	60.0%	60.0%	60.0%	53.7%	60.0%	60.0%	60.0%	60.0%	60.0%	60.0%

- For **MLPs** with arbitrary input embeddings, we perform a grid-search on the hyperparameters as follows:

  `num_layers` in $[2, 3, 4]$, `hidden_dimension` in $[128, 256, 512]$, and `dropout` in $[0.5, 0.6, 0.7]$.

  For both GNNs and MLPs across experimental datasets, the `learning_rate` is consistently set to $1e-2$, following previous studies (He et al., 2023; Li et al., 2024c). The total number of epochs is set to 500 with a patience of 100.

- For **SenBERT-66M** and **RoBERTa-355M**, we set the training epochs to 10 for semi-supervised settings and 4 for supervised settings. The `batch_size` is set to 32, and the `learning_rate` is set to $2e-5$.

- For **ENGINE** (Zhu et al., 2024b), we refer to the hyperparameter settings outlined in the original paper to determine the hyperparameter search space as follows:

  `num_layers` in $[1, 2, 3]$, `hidden_dimension` in $[64, 128]$, and `learning_rate` in $[5e-4, 1e-3]$.

  The neighborhood sampler is set to "Random Walk" for Cora while "k-Hop" with $k = 2$ for the remaining datasets.

- For **TAPE** (He et al., 2023), we utilize the provided prompt templates to guide Mistral-7B and GPT-4o in conducting reasoning. The LM, RoBERTa-355M, is fine-tuned based on its default parameter settings, while the GNN hyperparameters are explored with `num_layers` in $[2, 3, 4]$, `hidden_dimension` in $[128, 256]$, and with or without `batch_normalization`.

- For **LLM Instruction Tuning**, we use the LoRA (Hu et al., 2021) techniques to fine-tune LLMs. The `lora_r` parameter (dimension for LoRA update matrices) is set to 8 and the `lora_alpha` (scaling factor) to 16. The `dropout` ratio is set to 0.1, the `batch_size` to 16, and the `learning_rate` to $1e-5$. For each dataset, the input consists of the node's original text along with a carefully crafted task prompt designed to guide the LLMs in performing

the classification task. The expected output is the corresponding label. For small-scale datasets such as Cora, Citeseer, and Instagram, the number of training epochs is 10 in semi-supervised settings and 4 in supervised settings. For the remaining datasets, the training epochs are 2 and 1 for semi-supervised and supervised settings, respectively. The maximum input and output lengths are determined based on the average token lengths of each dataset.

- For **LLaGA** (Chen et al., 2024a), we empirically find that the HO templates consistently outperform the ND templates. Therefore, we set the HO templates as the default configuration, with `num_hop` set to 4. We use the text encoder as RoBERTa-355M. The linear projection layer $\phi_\theta(\cdot)$ consists of a $2-$layer MLP with a `hidden_dimension` of 2048. The `batch_size` is set to 64 and `learning_rate` to $1e-4$. The number of training epochs is set to 10 for semi-supervised settings and 4 for supervised settings. For Qwen2.5-series, we encounter over-fitting issues in the Photo, Computer, and Books datasets, leading us to adjust the learning rate to $5e-5$ and reduce the number of epochs to 2 under supervised settings.

- For **GraphGPT** (Tang et al., 2023), it includes three distinct stages: (1) text-graph grounding, (2) self-supervised instruction tuning, and (3) task-specific instruction tuning. Our empirical findings indicate that the inclusion of stage (1) does not consistently lead to performance improvements, thereby rendering this stage optional. For stage (2), we construct self-supervised training data for each dataset to perform dataset-specific graph matching tasks, adhering to the provided data format[2]. In stage (3), we utilize the training data to create $\langle$instruction, ground-truth label$\rangle$ pairs following the original prompt design. The training parameters for stage (2) include 2 epochs with a `learning_rate` of $1e-4$ and a `batch_size` of 16. For stage (3), we train for 10 epochs in semi-supervised settings and 6 epochs in supervised settings, with a `batch_size` of 32. Additionally, we adjust the maximum input and output lengths for each stage based on the dataset's text statistics.

- For **LLM Direct Inference**, we adopt two distinct categories of prompt templates: (1) advanced prompts that enhance the reasoning capabilities of LLMs, and (2) prompts enriched with structural information. These templates are illustrated in Appendix B.2 and strictly adhere to the zero-shot setting.

- For **ZeroG**, we adhere to its original parameter configurations by setting $k = 2$, the number of SGC iterations to 10, and the `learning_rate` to $1e-4$. In experiments involving **GFMs**, the intra-domain training mode utilizes the following source-target pairs: arXiv $\rightarrow$ Cora, arXiv $\rightarrow$ WikiCS, Reddit $\rightarrow$ Instagram, and Computer $\rightarrow$ Photo.

C.3. Distinct Features

A fair comparison necessitates a benchmark that evaluates all methods using consistent dataloaders, learning paradigms, backbone architectures, and implementation codebases. Our LLMNodeBed carefully follows these guidelines to support systematic and comprehensive evaluation of LLM-based node classification algorithms. Unlike existing benchmarks (Li et al., 2024c), which primarily rely on each algorithm's official implementation, LLMNodeBed distinguishes itself in the following ways:

- **Systematical Implementation:** We consolidate common components (e.g., DataLoader, Evaluation, Backbones) across algorithms to avoid code redundancy and enable fair comparisons and streamlined deployment. For example, several official implementations involve extensive code snippets, and we have produced cleaner, more streamlined versions that enhance both readability and usability. This systematic approach makes LLMNodeBed easily extendable to new datasets or algorithms.

- **Flexible Selection of Backbones:** LLMNodeBed incorporates a diverse selection of GNNs, LMs, and LLMs, which can be seamlessly integrated as components in baseline methods.

  - **GNNs:** Our framework supports a wide range of variants, including GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), GIN (Xu et al., 2019), and Graph Transformers (Shi et al., 2020). These GNNs can be customized with various layers and embedding dimensions.
  - **LMs and LLMs:** Open-source models can be easily loaded via the Transformers library[3]. In our experiments, we primarily utilize SenBERT-66M (Reimers & Gurevych, 2019), RoBERTa-355M (Liu et al., 2019), Qwen2.5-Series (Yang et al., 2024), Mistral-7B (Jiang et al., 2023), and LLaMA3.1-8B (Dubey et al., 2024). For close-source

[2] https://huggingface.co/datasets/Jiabin99/graph-matching
[3] https://huggingface.co/docs/transformers/en/index

LLMs, we have formatted the invocation functions of DeepSeek-Chat (Liu et al., 2024a) and GPT-4o (OpenAI, 2024). Additionally, LLMNodeBed allows users to specify and invoke any LM or LLM of their choice, providing flexibility for diverse research needs.

- **Robust Evaluation Protocols:** LLMNodeBed incorporates comprehensive hyperparameter tuning and design space exploration to fully leverage the potential of the algorithms. For instance, recent research (Luo et al., 2024b) highlights that classic GNNs remain strong baselines for node classification tasks, especially when the design space is expanded through techniques like residual connections, jumping knowledge, and selectable batch normalization. LLMNodeBed supports these enhancements, enabling the full utilization of GNNs. Furthermore, we conduct multiple experimental runs to enhance reliability and account for variability, which was often overlooked in previous studies.

D. Supplementary Materials for Comparisons among Algorithm Categories

Table 10: **Performance comparison under semi-supervised and supervised settings with Macro-F1 (%) reported.** The **best** and second-best results are highlighted. LLM_{IT} on the arXiv dataset requires extensive training time, preventing repeated experiments.

Semi-supervised		Cora	Citeseer	Pubmed	WikiCS	Instagram	Reddit	Books	Photo	Computer	Avg.
Classic	$\text{GCN}_{\text{ShallowEmb}}$	$80.76_{\pm 0.30}$	$66.00_{\pm 0.17}$	**$79.00_{\pm 0.30}$**	$77.87_{\pm 0.18}$	$52.44_{\pm 1.02}$	$61.15_{\pm 0.56}$	$22.18_{\pm 1.12}$	$62.65_{\pm 1.46}$	$61.33_{\pm 2.55}$	62.60
	$\text{SAGE}_{\text{ShallowEmb}}$	$80.88_{\pm 0.47}$	$65.06_{\pm 0.32}$	$77.88_{\pm 0.41}$	$77.02_{\pm 0.59}$	$50.74_{\pm 0.26}$	$56.39_{\pm 0.44}$	$24.37_{\pm 0.72}$	$74.17_{\pm 0.32}$	$71.67_{\pm 0.61}$	64.24
	$\text{GAT}_{\text{ShallowEmb}}$	$79.65_{\pm 1.03}$	$64.82_{\pm 1.20}$	$78.43_{\pm 0.73}$	$77.57_{\pm 1.01}$	$40.19_{\pm 1.56}$	$60.37_{\pm 1.22}$	$28.93_{\pm 3.65}$	$75.89_{\pm 0.60}$	$77.06_{\pm 2.98}$	64.77
	SenBERT-66M	$64.54_{\pm 1.18}$	$56.65_{\pm 1.40}$	$32.08_{\pm 3.35}$	$74.97_{\pm 0.96}$	$55.47_{\pm 0.76}$	$55.75_{\pm 0.58}$	$43.15_{\pm 1.53}$	$66.08_{\pm 0.76}$	$59.79_{\pm 1.30}$	56.50
	RoBERTa-355M	$70.41_{\pm 0.83}$	$63.36_{\pm 1.75}$	$40.82_{\pm 2.05}$	$73.98_{\pm 1.72}$	$57.43_{\pm 0.42}$	$59.23_{\pm 0.36}$	**$51.34_{\pm 0.95}$**	$67.92_{\pm 0.49}$	$63.38_{\pm 2.17}$	60.87
Encoder	$\text{GCN}_{\text{LLMEmb}}$	$81.19_{\pm 0.59}$	**$67.17_{\pm 0.73}$**	$78.39_{\pm 0.36}$	$78.58_{\pm 0.51}$	**$58.97_{\pm 0.85}$**	$68.46_{\pm 0.91}$	$39.64_{\pm 0.85}$	$79.87_{\pm 0.57}$	$77.36_{\pm 0.70}$	69.95
	ENGINE	**$82.52_{\pm 0.45}$**	**$67.15_{\pm 0.15}$**	$77.53_{\pm 0.34}$	$78.89_{\pm 0.38}$	$57.25_{\pm 2.50}$	**$69.56_{\pm 0.21}$**	$34.04_{\pm 1.10}$	$78.55_{\pm 1.12}$	$75.86_{\pm 0.60}$	69.04
Reasoner	TAPE	$81.89_{\pm 0.31}$	$66.80_{\pm 0.33}$	$78.46_{\pm 1.13}$	**$80.03_{\pm 0.23}$**	$50.01_{\pm 1.60}$	$61.23_{\pm 0.69}$	$47.12_{\pm 3.26}$	**$82.31_{\pm 0.19}$**	**$84.90_{\pm 1.14}$**	**70.31**
Predictor	LLM_{IT}	$56.35_{\pm 0.22}$	$47.34_{\pm 0.68}$	$62.81_{\pm 0.21}$	$65.75_{\pm 0.17}$	$38.30_{\pm 0.94}$	$44.41_{\pm 8.86}$	$39.44_{\pm 0.44}$	$60.71_{\pm 0.09}$	$57.38_{\pm 0.65}$	52.50
	GraphGPT	$58.33_{\pm 0.81}$	$54.21_{\pm 1.11}$	$56.09_{\pm 0.88}$	$62.04_{\pm 0.62}$	$38.78_{\pm 0.60}$	$38.88_{\pm 0.28}$	$42.85_{\pm 0.94}$	$65.77_{\pm 1.34}$	$66.69_{\pm 1.49}$	53.74
	LLaGA	$71.14_{\pm 4.47}$	$52.53_{\pm 3.59}$	$45.12_{\pm 7.63}$	$70.48_{\pm 2.94}$	$50.12_{\pm 10.45}$	$54.67_{\pm 11.24}$	$39.70_{\pm 2.44}$	$79.32_{\pm 2.42}$	$78.01_{\pm 1.36}$	60.12

Supervised		Cora	Citeseer	Pubmed	arXiv	WikiCS	Instagram	Reddit	Books	Photo	Computer	Avg.
Classic	$\text{GCN}_{\text{ShallowEmb}}$	$86.54_{\pm 1.44}$	$71.52_{\pm 1.71}$	$88.54_{\pm 0.60}$	$50.28_{\pm 0.84}$	$82.11_{\pm 0.61}$	$54.91_{\pm 0.48}$	$65.00_{\pm 0.42}$	$34.39_{\pm 1.26}$	$66.04_{\pm 2.85}$	$64.60_{\pm 4.99}$	66.39
	$\text{SAGE}_{\text{ShallowEmb}}$	$86.37_{\pm 1.26}$	$71.87_{\pm 1.34}$	$90.16_{\pm 0.27}$	$49.73_{\pm 0.49}$	$82.78_{\pm 1.53}$	$51.37_{\pm 1.67}$	$61.39_{\pm 0.54}$	$38.29_{\pm 2.54}$	$80.37_{\pm 0.34}$	$82.93_{\pm 0.49}$	69.53
	$\text{GAT}_{\text{ShallowEmb}}$	$85.64_{\pm 0.87}$	$69.27_{\pm 2.15}$	$87.70_{\pm 0.48}$	$49.71_{\pm 0.23}$	$82.14_{\pm 1.04}$	$50.26_{\pm 3.16}$	$64.11_{\pm 1.06}$	$42.85_{\pm 1.62}$	$80.82_{\pm 0.89}$	$84.74_{\pm 0.79}$	69.72
	SenBERT-66M	$77.13_{\pm 2.19}$	$71.25_{\pm 1.03}$	$93.95_{\pm 0.39}$	$52.48_{\pm 0.12}$	$84.43_{\pm 1.58}$	$56.12_{\pm 0.66}$	$58.31_{\pm 0.76}$	$52.96_{\pm 1.78}$	$70.39_{\pm 0.54}$	$65.08_{\pm 0.37}$	68.21
	RoBERTa-355M	$81.38_{\pm 1.17}$	$72.31_{\pm 1.45}$	**$94.33_{\pm 0.14}$**	$57.25_{\pm 0.53}$	**$86.10_{\pm 1.11}$**	$59.10_{\pm 1.38}$	$60.16_{\pm 0.94}$	**$57.24_{\pm 1.27}$**	$72.89_{\pm 0.50}$	$70.64_{\pm 0.58}$	71.14
Encoder	$\text{GCN}_{\text{LLMEmb}}$	**$87.23_{\pm 1.34}$**	$72.71_{\pm 0.86}$	$87.76_{\pm 0.76}$	$55.22_{\pm 0.46}$	$82.78_{\pm 1.68}$	**$60.38_{\pm 0.53}$**	$70.64_{\pm 0.75}$	$48.18_{\pm 2.29}$	$80.51_{\pm 0.65}$	$85.10_{\pm 1.00}$	73.05
	ENGINE	$85.72_{\pm 1.58}$	$71.22_{\pm 2.17}$	$89.63_{\pm 0.14}$	$56.32_{\pm 0.60}$	$83.80_{\pm 1.06}$	$60.02_{\pm 1.26}$	**$71.17_{\pm 0.75}$**	$48.24_{\pm 2.64}$	$82.77_{\pm 0.28}$	$84.15_{\pm 0.84}$	73.30
Reasoner	TAPE	$87.21_{\pm 1.60}$	**$73.33_{\pm 1.57}$**	$92.39_{\pm 0.02}$	$57.79_{\pm 0.49}$	$86.03_{\pm 1.14}$	$58.31_{\pm 1.15}$	$65.91_{\pm 0.71}$	$54.07_{\pm 2.01}$	$83.41_{\pm 0.42}$	$86.78_{\pm 0.53}$	**74.52**
Predictor	LLM_{IT}	$66.93_{\pm 4.54}$	$52.22_{\pm 2.71}$	$93.45_{\pm 0.25}$	57.48	$78.39_{\pm 1.20}$	$42.15_{\pm 4.54}$	$56.65_{\pm 0.85}$	$49.86_{\pm 0.71}$	$68.74_{\pm 2.54}$	$62.78_{\pm 2.83}$	62.86
	GraphGPT	$74.08_{\pm 4.36}$	$61.04_{\pm 1.24}$	$80.98_{\pm 11.22}$	$56.80_{\pm 0.08}$	$73.92_{\pm 0.61}$	$40.07_{\pm 2.10}$	$39.97_{\pm 1.77}$	$47.97_{\pm 1.94}$	$74.22_{\pm 0.43}$	$74.19_{\pm 1.75}$	62.32
	LLaGA	$84.97_{\pm 3.97}$	$72.59_{\pm 1.70}$	$90.00_{\pm 0.80}$	**$58.08_{\pm 0.29}$**	$82.37_{\pm 1.73}$	$57.96_{\pm 2.40}$	$62.14_{\pm 15.59}$	$54.89_{\pm 2.29}$	**$83.56_{\pm 0.40}$**	**$86.97_{\pm 0.34}$**	73.35

Table 11: **Performance comparison under zero-shot setting with both Accuracy (%) and Macro-F1 (%) reported.**

Type & LLM	Method	**Cora** (82.52)		**WikiCS** (68.67)		**Instagram** (63.35)		**Photo** (78.50)		**Avg.**	
		Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
LLM DeepSeek-Chat	Direct	68.06	60.07	71.41	65.21	42.42	39.42	65.25	56.92	61.78	55.40
	CoT	68.08	60.32	71.83	60.73	**43.47**	28.22	63.97	57.61	61.84	51.72
	ToT	66.61	59.10	57.35	54.68	36.95	23.94	59.25	54.39	55.04	48.03
	ReAct	65.68	58.68	71.24	60.97	43.30	28.42	63.66	56.23	60.97	51.08
	w. Neighbor	68.63	**69.21**	70.26	64.26	43.34	**41.57**	61.57	54.84	60.95	57.47
	w. Summary	**73.62**	64.80	**72.53**	**67.28**	41.18	37.56	**72.73**	**70.58**	**65.02**	**60.05**
LLM Mistral-7B	Direct	59.65	58.34	70.13	67.80	44.29	42.16	**57.54**	55.50	57.90	55.95
	CoT	58.02	57.13	69.00	66.17	45.48	44.56	49.56	51.42	55.52	54.82
	ToT	58.78	57.20	67.56	64.52	45.39	44.73	44.25	46.87	54.00	53.33
	ReAct	60.32	60.89	**71.02**	67.31	**46.26**	**46.09**	52.47	50.92	57.52	56.30
	w. Neighbor	67.69	66.62	68.32	65.58	37.05	28.23	53.39	56.06	56.61	54.12
	w. Summary	**68.12**	**67.45**	70.52	**67.87**	41.94	38.93	56.01	**56.22**	**59.15**	**57.62**

E. Supplementary Materials for Fine-grained Analysis

E.1. LLM-as-Encoder: Compared with LMs

Table 12: **Comparison of LLM- and LM-as-Encoder with Accuracy (%) reported under supervised setting. LLM-as-Encoder outperforms LMs in heterophilic graphs.** The **best encoder** within each method on a dataset is highlighted.

Method	Encoder	Computer	Cora	Pubmed	Photo	Books	Citeseer	WikiCS	arXiv	Instagram	Reddit
Homophily Ratio (%)		85.28	82.52	79.24	78.50	78.05	72.93	68.67	63.53	63.35	55.22
MLP	SenBERT	71.75±0.20	75.72±2.19	90.26±0.37	74.59±0.19	83.61±0.43	69.84±2.21	81.13±1.01	68.60±0.16	67.12±1.01	58.40±0.56
	RoBERTa	**72.36±0.09**	80.92±2.65	90.54±0.37	75.50±0.25	**84.06±0.37**	**74.11±1.33**	83.18±0.96	73.65±0.11	68.57±0.84	61.06±0.31
	Qwen-3B	69.25±0.34	81.29±1.05	92.02±0.38	74.35±0.47	83.43±0.56	72.88±1.66	85.46±0.84	74.62±0.19	68.62±0.54	61.39±0.36
	Mistral-7B	71.38±0.17	**81.62±0.63**	**92.73±0.24**	**75.83±0.25**	83.96±0.46	73.85±1.75	**86.92±0.90**	**75.29±0.16**	**69.03±0.35**	**62.49±0.20**
GCN	SenBERT	**90.53±0.18**	**88.33±0.90**	**89.26±0.24**	**86.79±0.18**	**84.60±0.38**	75.31±0.55	84.28±0.26	73.29±0.22	68.13±0.18	69.04±0.46
	RoBERTa	90.16±0.13	87.96±1.98	89.00±0.21	86.79±0.48	84.42±0.38	**76.57±0.99**	84.64±0.27	74.13±0.19	**68.43±0.51**	69.28±0.50
	Qwen-3B	88.06±0.35	88.24±1.79	88.42±0.51	85.20±0.38	84.34±0.61	76.37±0.97	84.05±0.55	73.62±0.33	68.32±0.73	**71.04±0.42**
	Mistral-7B	89.52±0.31	88.15±1.79	88.38±0.68	86.07±0.63	84.23±0.20	76.45±1.19	**84.78±0.86**	**74.39±0.31**	68.27±0.45	70.65±0.75
SAGE	SenBERT	**90.86±0.18**	87.36±1.79	**90.93±0.13**	87.41±0.33	**85.13±0.27**	74.73±0.67	85.94±0.52	73.43±0.23	67.72±0.43	64.13±0.41
	RoBERTa	90.70±0.25	87.36±1.69	90.38±0.09	**87.42±0.51**	85.13±0.41	**75.90±0.41**	86.31±0.68	75.28±0.31	68.84±0.54	**64.85±0.31**
	Qwen-3B	87.44±0.66	**87.36±1.10**	89.98±0.38	85.17±0.44	84.69±0.31	75.63±0.94	85.58±0.58	75.20±0.49	68.43±0.57	63.98±0.69
	Mistral-7B	90.16±0.26	87.22±1.24	90.54±0.50	87.34±0.43	85.01±0.49	75.20±1.34	**87.87±0.35**	**76.18±0.34**	**69.39±0.52**	64.34±0.23

We supplement the comparison between LLM-as-Encoder and LM-as-Encoder under supervised settings in Table 12. The key takeaway that **LLMs outperform LMs as Encoders in heterophilic graphs** remains valid. This conclusion is particularly evident on the arXiv dataset, where the performance gap between LM- and LLM-generated embeddings reaches up to 7% on MLP and 3% on GraphSAGE. Additionally, we observe that in supervised settings, the performance gap between LM- and LLM-as-Encoders becomes less pronounced compared to semi-supervised settings. We still consider the theoretical insights in Equation (1) for explanation: Increased supervision enhances the mutual information between labels and graph structure, i.e., $I(\mathcal{E}, \mathcal{Y}_l)$, thereby rendering the second term less significant and diminishing the advantages provided by more powerful encoders like LLMs.

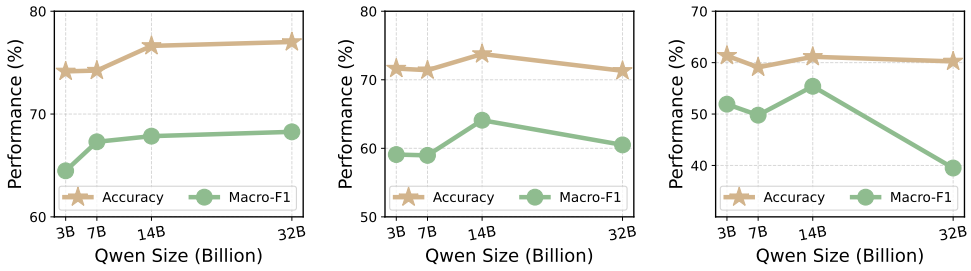
E.2. LLM-as-Predictor: Sensitivity to LLM Backbones

Table 13: **Sensitivity of LLaGA to different LLM backbones under semi-supervised Settings**.
The best LLM backbone within **each series** and **at similar scales** is highlighted.

		LLM	Cora	Citeseer	Pubmed	WikiCS	Instagram	Reddit	Books	Photo	Computer
Accuracy (%)	Same series	Qwen-3B	74.15±2.41	62.74±10.42	54.97±10.71	71.64±1.34	**61.35±2.35**	65.11±1.59	82.26±0.43	**83.85±0.77**	**85.84±1.12**
		Qwen-7B	74.23±1.58	64.79±1.77	**62.58±1.36**	71.40±2.28	59.09±3.32	66.07±0.32	81.38±1.58	80.75±1.60	84.47±1.73
		Qwen-14B	76.63±1.79	**66.06±1.86**	62.17±6.86	**73.76±0.42**	61.14±3.84	**66.59±1.23**	81.40±0.20	82.47±0.80	85.08±0.36
		Qwen-32B	**77.01±3.62**	64.01±2.59	58.60±7.93	71.31±3.05	60.24±4.03	66.19±2.11	**82.34±0.44**	82.85±1.52	85.74±1.65
	Similar scales	Mistral-7B	**78.94±1.14**	62.61±3.63	**65.91±2.09**	**76.47±2.20**	**65.84±0.72**	**70.10±0.38**	**83.47±0.45**	**84.44±0.90**	**87.82±0.53**
		Qwen-7B	74.23±1.58	**64.79±1.77**	62.58±1.36	71.40±2.28	59.09±3.32	66.07±0.32	81.38±1.58	80.75±1.60	84.47±1.73
		LLaMA-8B	75.34±1.09	61.33±2.11	61.84±3.62	72.15±3.32	55.77±3.07	65.09±1.39	81.30±0.07	82.26±1.67	86.43±0.25
Macro-F1 (%)	Same series	Qwen-3B	64.48±4.32	53.25±2.21	44.35±3.64	59.09±3.00	51.92±8.02	42.42±0.50	**37.56±2.56**	**70.94±0.97**	**70.65±3.31**
		Qwen-7B	67.30±5.70	53.43±1.63	**47.55±1.16**	58.97±3.67	49.78±8.13	**51.32±10.10**	34.33±4.37	66.94±4.64	69.23±3.26
		Qwen-14B	67.85±4.53	**55.39±3.99**	45.90±7.69	**64.10±0.24**	**55.40±1.26**	44.35±0.86	37.50±0.35	70.33±1.12	69.19±5.16
		Qwen-32B	**68.27±6.24**	53.11±2.59	46.52±7.54	60.50±3.40	39.50±8.07	43.96±1.45	35.30±1.80	70.08±2.39	67.26±3.89
	Similar scales	Mistral-7B	**71.14±4.47**	52.53±3.59	45.12±7.63	**70.48±2.94**	**50.12±10.45**	**54.67±11.24**	**39.70±2.44**	**79.32±2.42**	**78.01±1.36**
		Qwen-7B	67.30±5.70	**53.43±1.63**	47.55±1.16	58.97±3.67	49.78±8.13	51.32±10.10	34.33±4.37	66.94±4.64	69.23±3.26
		LLaMA-8B	67.50±4.73	51.22±1.27	**47.80±2.78**	64.17±6.00	48.56±6.92	43.31±0.94	34.49±1.48	72.45±0.35	71.43±4.43

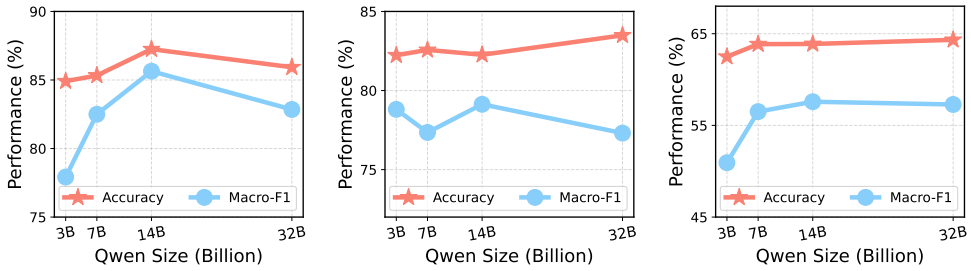
We evaluate LLaGA with different LLM backbones in semi-supervised settings, as detailed in Table 13. We examine two primary trends: (1) **Scaling within the same series**: Assessing whether scaling laws apply to node classification tasks by using LLMs from the same series, and (2) **Model selection at similar scales**: Identifying the most suitable LLM for node classification tasks by comparing models of similar scales.

Scaling within the same series: We plot performance trends across several datasets under both semi-supervised and supervised settings to clearly illustrate these dynamics. From Figure 3 and Figure 4, we conclude that scaling laws generally



(a) Cora (b) WikiCS (c) Instagram

Figure 3: **Performance trends within Qwen-series in different scales using LLaGA framework in semi-supervised settings.**



(a) Cora (b) WikiCS (c) Instagram

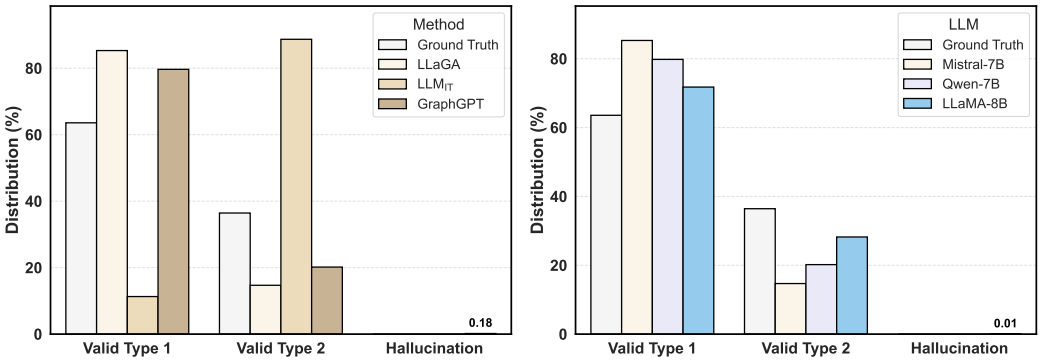
Figure 4: **Performance trends within Qwen-series in different scales using LLaGA framework in supervised settings.**

Table 14: **Training and inference times of Qwen-series models ranging from 3B to 32B parameters.**

GPU Device	LLM	Semi-supervised Training Times			Supervised Training Times			Avg. Inference Times Per Case		
		Cora	WikiCS	Instagram	Cora	WikiCS	Instagram	Cora	WikiCS	Instagram
1 NVIDIA A6000-48G	Qwen-3B	2.2min	7.1min	5.8min	8.6min	33.5min	20.2min	32.3ms	37.3ms	26.9ms
	Qwen-7B	4.7min	15.3min	8.2min	13.3min	59.4min	43.2min	50.9ms	55.9ms	43.4ms
2 NVIDIA A6000-48G	Qwen-14B	8.9min	25.5min	15.7min	30.8min	2.3h	1.5h	97.6ms	103.0ms	83.2ms
	Qwen-32B	18.9min	43.6min	30.7min	52.2min	3.3h	2.0h	254.7ms	262.6ms	232.4ms

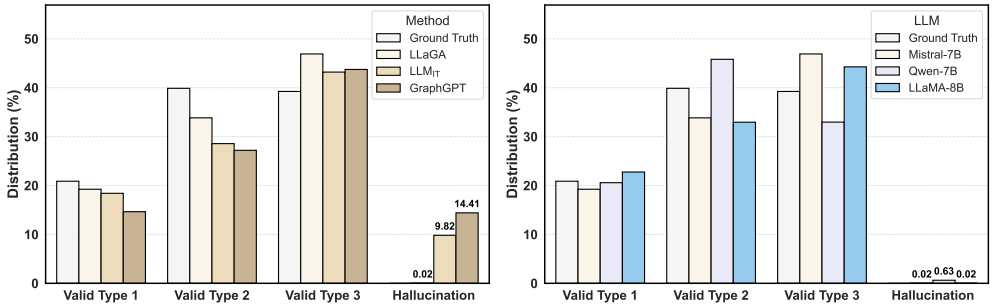
hold: as the Qwen model size increases from 3B to 32B parameters, performance improves, indicating that larger model sizes enhance task performance. However, the 7B and 14B models are sufficiently large, typically representing the point beyond which further increases in model size yield only marginal improvements but introducing huge computational costs (Table 14). Unexpectedly, in the Instagram dataset under semi-supervised settings, the Qwen-32B model experiences a performance drop. This may be because 32B models require extensive data to train effectively, making them less robust and stable compared to smaller models. Based on these findings, we recommend the 7B or 14B models as they offer an optimal balance between performance and computational costs.

Model selection at similar scales: By comparing the performance of Mistral-7B, Qwen-7B, and LLaMA-8B in Table 13, we conclude that Mistral-7B outperforms the other two similarly scaled LLMs in most cases. This makes Mistral-7B the optimal choice as a backbone LLM for node classification tasks.



(a) Different LLM-as-Predictor Methods (b) Different LLM Backbones within LLaGA

Figure 5: **Biased predictions by LLM-as-Predictor methods on the Instagram dataset:** Comparison of ground-truth label distributions with predictor-generated label distributions.



(a) Different LLM-as-Predictor Methods (b) Different LLM Backbones within LLaGA

Figure 6: **Biased predictions by LLM-as-Predictor methods on the Pubmed dataset.**

E.3. LLM-as-Predictor: Biased and Hallucinated Predictions

During our experiments, we found that LLM-as-Predictor methods are vulnerable to limited supervision. In addition to standard metrics such as Accuracy (Table 1) and Macro-F1 scores (Table 10), their predictions also exhibit significant **biases** and **hallucinations**.

Biased Predictions: For datasets with fewer labels, LLM-as-Predictor methods tend to disproportionately predict certain labels while under-predicting others. To illustrate this phenomenon, we compare the ground-truth label distributions with the predicted label distributions. Specifically, we present different LLM-as-Predictor methods, LLM$_{\text{IT}}$, GraphGPT, and LLaGA, in Figure 5a, and the LLaGA method with various LLM backbones in Figure 5b, using the Instagram dataset in semi-supervised settings, which has two labels.

From Figure 5a, we can directly observe that LLaGA and GraphGPT predominantly bias towards the first class, while LLM$_{\text{IT}}$ tends to predict the second class more frequently. The predicted label distributions of LLMs are more **polarized** compared to the ground-truth distributions, where the two labels are roughly in a $6:4$ proportion. In contrast, LLMs tend to predict in ratios such as $8:2$ or $1:9$. This observation also holds across different LLMs, as shown in Figure 5b, where both Qwen-7B and LLaMA-8B tend to bias towards the first label. A similar example on the Pubmed dataset, which contains three classes in a semi-supervised setting, is shown in Figure 6. Here, the predictor methods tend to bias towards the third class, while LLaGA with Qwen-7B tends to predict the second class. Additionally, in the semi-supervised setting for Pubmed, the training data consists of only 60 samples, which is insufficient to train a robust predictor model, leading to high levels of hallucinations across all methods.

Hallucinations: In addition to biased predictions, we observed that a certain portion of the LLMs' outputs **fall outside the valid label space** or **contain unexpected content that cannot be parsed**. In semi-supervised settings, the limited training data restricts these predictor methods from developing effective models, resulting in failures to follow instructions

Table 15: **Average hallucination ratios (%) of LLM-as-Predictor methods.** The hallucination rate is calculated as the proportion of outputs containing invalid labels or unexpected content across all test cases, where higher values indicate poorer classification ability. Hallucinations $> 1\%$ are **highlighted**.

Setting	Method	Cora	Citeseer	Pubmed	WikiCS	Instagram	Reddit	Books	Photo	Computer
Semi-supervised	# Train Samples	140	120	60	580	1,160	3,344	4,155	4,836	8,722
	LLM_{IT}	0.43	**13.08**	**9.24**	0.06	0.00	0.00	0.01	0.02	0.02
	GraphGPT	**7.56**	**2.51**	**15.97**	**7.76**	0.28	**1.78**	0.51	0.72	0.27
	LLaGA	0.35	0.20	0.29	0.00	0.01	0.02	0.00	0.00	0.00
Supervised	# Train Samples	1,624	1,911	11,830	7,020	6,803	20,060	24,930	29,017	52,337
	LLM_{IT}	0.06	**13.61**	0.00	0.00	0.00	0.00	0.00	0.01	0.02
	GraphGPT	**1.29**	0.63	0.08	**1.64**	0.13	0.50	0.11	0.11	0.10
	LLaGA	0.03	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00

Table 16: **Examples of hallucinations in GraphGPT's outputs on the Citeseer and Pubmed datasets.**

Error Type	Citeseer		Pubmed	
	Prediction	Ground-truth	Prediction	Ground-truth
Misspelling	AGents	Agents	Type II diabetes	Type 2 diabetes
Non-existent Types	Logic and Mathematics	ML (Machine Learning)	Type 3 diabetes of the young (MODY)	Type 2 diabetes
	Information Extraction	IR (Information Retrieval)	Genetic Studies of Wolfram Syndrome	Type 2 diabetes
	Pattern Recognition	ML (Machine Learning)	Experimentally induced insulin resistance	Type 2 diabetes
	Multiagent Systems	Agents	Experimentally induced oxidative stress	Experimentally induced diabetes
Unexpected Contents	H.4.1 Office Automation: Workflow Management The citation graph is given by the following: ...		membrane is not altered by diabetes. What is the sensitivity and specificity of the IgA-EMA test ...	

and understand dataset-specific classification requirements. To illustrate that, we provide both quantitative and qualitative analyses as follows:

- **Quantitative Analysis:** Table 15 presents the hallucination rates of each LLM-as-Predictor method across various experimental datasets in both semi-supervised and supervised settings. The hallucination rate is calculated as the proportion of outputs containing invalid labels or unexpected content among all test cases, where higher values indicate poorer classification performance. Hallucinations are most severe on the Pubmed and Citeseer datasets within semi-supervised settings, where the number of training samples does not exceed hundreds, making effective model training challenging. **This demonstrates that the number of training samples significantly impacts the mitigation of hallucinations**: even in semi-supervised settings, larger datasets like Books and Photo provide thousands of training samples, resulting in hallucination ratios consistently below 1%. Therefore, this further verifies that LLM-as-Predictor methods require extensive labeled data for effective model training.
- **Qualitative Analysis:** We provide several examples to facilitate the comprehension of hallucinated predictions, which we categorize into three types: (1) misspellings of existing labels, (2) generation of non-existent types, and (3) unexpected content that cannot be parsed. Illustrative examples of these types from GraphGPT's outputs on the Citeseer and Pubmed datasets are presented in Table 16.

F. Supplementary Materials for Computational Cost Analysis

Table 17: **Total training times of different methods in semi-supervised settings.** All recorded experiment times are based on a single NVIDIA H100-80G GPU.

Type	Method	Cora	Citeseer	Pubmed	WikiCS	Instagram	Reddit	Books	Photo	Computer
# Training Samples		140	120	60	580	1,160	3,344	4,155	4,836	8,722
Classic	$GCN_{ShallowEmb}$	2.8s	2.7s	2.7s	3.2s	1.8s	7.7s	12.7s	13.8s	33.5s
	$GAT_{ShallowEmb}$	1.9s	2.4s	3.8s	3.3s	2.0s	6.0s	10.5s	12.3s	38.0s
	$SAGE_{ShallowEmb}$	1.9s	3.9s	5.0s	2.9s	1.8s	6.4s	16.9s	21.1s	33.3s
	SenBERT-66M	8.5s	7.9s	5.9s	27.9s	14.7s	1.2m	1.5m	1.8m	3.3m
	RoBERTa-355M	21.2s	18.9s	12.7s	1.2m	2.3m	6.5m	8.1m	3.8m	6.9m
Encoder	GCN_{LLMEmb}	1.2m	1.4m	13.4m	7.4m	4.5m	16.0m	23.5m	26.8m	44.7m
	ENGINE	2.2m	2.4m	16.1m	15.2m	9.3m	22.9m	31.1m	38.8m	1.1h
Reasoner	TAPE	25.5m	27.8m	5.6h	2.7h	2.0h	8.0h	9.9h	11.7h	14.5h
Predictor	LLM_{IT}	25.6m	22.0m	3.9m	1.1h	49.1m	1.1m	2.0h	2.4h	2.7h
	GraphGPT	16.4m	15.5m	1.2h	48.5m	30.1m	1.8h	2.4h	2.2h	5.8h
	LLaGA	1.7m	2.2m	5.2m	5.8m	3.0m	20.9m	19.5m	23.5m	43.1m

Table 18: **Total training times of different methods in supervised settings.** All recorded experiment times are based on a single NVIDIA H100-80G GPU.

Type	Method	Cora	Citeseer	Pubmed	WikiCS	arXiv	Instagram	Reddit	Books	Photo	Computer
# Training Samples		1,624	1,911	11,830	7,020	90,941	6,803	20,060	24,930	29,017	52,337
Classic	$GCN_{ShallowEmb}$	1.8s	1.7s	5.2s	5.1s	51.2s	19.5s	8.5s	14.9s	19.7s	25.8s
	$GAT_{ShallowEmb}$	2.1s	1.9s	7.9s	5.7s	1.5m	2.7s	6.9s	16.6s	28.0s	44.6s
	$SAGE_{ShallowEmb}$	1.7s	3.0s	7.6s	4.0s	1.3m	2.0s	7.2s	19.6s	20.1s	43.2s
	SenBERT-66M	35s	41s	2.6m	2.5m	7.4m	1.2m	4.4m	1.8m	2.2m	4.1m
	RoBERTa-355M	1.3m	1.6m	9.2m	5.5m	40.8m	5.3m	15.9m	9.7m	11.9m	22.4m
Encoder	GCN_{LLMEmb}	1.2m	1.4m	13.4m	7.5m	1.4h	4.5m	16.1m	23.6m	26.8m	44.8m
	ENGINE	2.2m	2.4m	16.1m	19.4m	2.6h	8.9m	24.2m	35.2m	44.2m	1.2h
Reasoner	TAPE	27.4m	30.3m	5.9h	2.8h	37.4h	2.1h	8.3h	10.0h	12.0h	15.0h
Predictor	LLM_{IT}	1.0h	1.3h	9.9h	4.2h	36.3h	2.7h	3.4h	5.7h	7.4h	12.4h
	GraphGPT	26.4m	29.5m	2.7h	1.7h	7.8h	49.1m	3.4h	3.8h	3.6h	7.8h
	LLaGA	5.6m	7.7m	25.6m	18.8m	7.7h	10.6m	32.2m	1.0h	1.4h	2.5h

Table 19: **Inference times of different methods.** Values in brackets denote the average inference time per case in milliseconds (ms). All recorded experiment times are based on a single NVIDIA H100-80G GPU.

Method		Cora	arXiv	Instagram	Photo	WikiCS
# Test Samples		542	48,603	5,847	2,268	9,673
Classic	GCN	0.9ms	21.8ms	2.0ms	7.5ms	4.4ms
Encoder	GCN_{LLMEmb}	14.0s (26ms)	23.8m (29ms)	53.6s (24ms)	5.3m (33ms)	3.7m (38ms)
Reasoner	TAPE	5.0m (551ms)	10.4h (767ms)	23.7m (627ms)	2.3h (863ms)	1.3h (813ms)
Predictor	LLM_{IT}	1.2m (129ms)	3.3h (243ms)	2.7m (71ms)	24.1m (149ms)	5.8m (60ms)
	GraphGPT	1m (104ms)	1.2h (87ms)	2.0m (52ms)	10.4m (64ms)	11.0m (112ms)
	LLaGA	11.2s (21ms)	57.1m (70ms)	1.3m (35ms)	4.4m (27ms)	2.4m (25ms)