GNN-as-Judge: Unleashing the Power of LLMs for Graph Semi-Supervised Learning with GNN Feedback

Ruiyao Xu
Northwestern University
Evanston, IL, USA
ruiyaoxu2028@u.northwestern.edu

Kaize Ding
Northwestern University
Evanston, IL, USA
kaize.ding@northwestern.edu

Abstract

Large Language Models (LLMs) have shown strong performance on text-attributed graphs (TAGs) due to their superior semantic understanding ability on textual node features. However, their effectiveness in the semi-supervised setting, where labeled nodes are rather limited, remains constrained since fine-tuning LLMs usually requires sufficient labeled data, especially when the TAG shows complex structural patterns. In essence, this paper targets two key challenges: (i) the difficulty of generating reliable pseudo labels on TAGs for LLMs, and (ii) the need to mitigate potential label noise when fine-tuning LLMs with pseudo labels. To counter the challenges, we propose a new framework, *GNN-as-Judge*, which can unleash the power of LLMs for semi-supervised learning on TAGs by incorporating the structural inductive bias of Graph Neural Networks (GNNs). Specifically, *GNN-as-Judge* introduces a collaborative pseudo-labeling strategy that exploits both the agreement and disagreement between LLMs and GNNs, and a weakly-supervised LLM fine-tuning algorithm that can distill the knowledge from informative pseudo labels while mitigating the potential label noise. Experiments on different TAG datasets demonstrate that GNN-as-Judge significantly outperforms existing methods, especially under low-resource regimes.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; • **Mathematics of computing** → **Graph algorithms**.

Keywords

Semi-supervised Learning, Graph Neural Networks, Large Language Models

ACM Reference Format:
Ruiyao Xu and Kaize Ding. 2025. GNN-as-Judge: Unleashing the Power of LLMs for Graph Semi-Supervised Learning with GNN Feedback. In *Proceedings of Machine Learning on Graphs in the Era of Generative Artificial Intelligence (MLoG-GenAI@KDD '25).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/XXXXXXX.XXXXXXX

MLoG-GenAI@KDD '25, Toronto, Canada

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/10.1145/XXXXXXX.XXXXXXX

1 Introduction

Text-Attributed Graphs (TAGs), where nodes correspond to text documents and edges represent their relationships, are pervasive across various applications such as citation networks, social media platforms, and e-commerce ecosystems [7, 21, 52]. Unlike conventional attributed graphs, TAGs encode raw textual contents rather than numerical values, which requires more dedicated mechanisms to effectively capture semantic information while preserving structural relationships. Recent advances in Large Language Models (LLMs) have shown exceptional capabilities in text understanding [6, 12, 35], driving growing interest in leveraging their exceptional text understanding capabilities to address TAG-related tasks [9, 52].

It is noteworthy that current research of LLMs for node classification on TAGs primarily focuses on the *supervised setting* where abundant labeled data is accessible, mainly due to the reason that fine-tuning LLMs on TAGs requires sufficient supervision signals to align the graph and text token spaces [8, 47, 56]. However, real-world graphs are usually sparsely labeled, directly applying existing methods to such *semi-supervised setting* will easily lead to overfitting and poor generalization [47, 52, 56]. Although one can leverage pseudo-labeling techniques to augment the limited labeled training data with unlabeled data [29, 41], it remains unclear how to solve the following two critical challenges in LLMs for semi-supervised node classification on TAGs:

- *C1: How to go beyond the knowledge of LLMs and obtain reliable pseudo-labeled data?* Since LLMs are inherently difficult to interpret complex graph structural patterns [17, 24], solely relying on LLMs with no structural inductive bias to generate pseudo labels is less desirable. Although text-based serialization methods try to encode structural context into the prompts, those methods could still struggle with self-generated pseudo labels due to the hallucination and self-bias of LLMs [9, 14, 31, 32]. Generating reliable pseudo-labeled examples that encode not only textual but also structural inductive bias is therefore crucial for enabling LLMs to transcend their inherent knowledge limitations on graphs.
- *C2: How to extract the knowledge from pseudo-labeled data while mitigating the potential label noise during LLM fine-tuning?* Despite pseudo-labeling showing its empirical effectiveness in many fields, the potential label noise has been a longstanding problem. Especially for those "hard" pseudo-labeled examples that are more valuable for training the model, they could also bring more label noise if the labels are incorrect. Simply performing LLM supervised fine-tuning with the noisy pseudo labels can lead to performance degradation [10, 26, 45], which necessitates the need to develop a new learning algorithm that can effectively distill the knowledge and mitigate the label noise when fine-tuning with pseudo-labeled data.

In this paper, we propose *GNN-as-Judge*, a novel framework that fine-tunes LLMs on sparsely labeled graphs using feedback from GNNs. Instead of mining the easy and hard pseudo-labeled nodes solely based on LLM confidence, at its core, a GNN with structural inductive bias acts as a judge to provide additional guidance for LLM to generate reliable pseudo-labels. *GNN-as-Judge* strategically leverages both agreement and disagreement between GNN and LLM as signals to identify not only "easy", but more remarkably, "hard" pseudo labels that LLMs are more likely to make mistakes. To further mitigate the potential label noise, especially in the harder examples, we develop a weakly-supervised LLM fine-tuning algorithm that jointly performs fine-tuning on the two selected pseudo-labeled node sets. Specifically, in addition to applying supervised LLM instruction tuning on the agreement (easy) node set, we propose to conduct preference tuning on the disagreement (hard) node set, which allows LLM to learn relative preferences between predictions from the two models. In this manner, LLM can effectively leverage additional supervision signals that encode structural inductive biases from GNN while circumventing overfitting to incorrect pseudo labels. To summarize, our contributions are mainly three-folds:

- We study the problem of LLMs for graph semi-supervised learning, a fundamental but underexplored research problem, where the key challenges lie in selecting reliable pseudo labels and mitigating label noise during fine-tuning.
- We propose *GNN-as-Judge*, a novel framework that positions GNNs as judges to select reliable pseudo labels for fine-tuning LLMs. *GNN-as-Judge* is also equipped with a new weakly-supervised fine-tuning algorithm that can further mitigate label noise during LLM fine-tuning.
- We conduct comprehensive experiments on different TAG datasets with various scales. Results demonstrate that *GNN-as-Judge* significantly outperforms both traditional GNN-based approaches and other LLM-based baselines, especially in extreme low-resource scenarios.

2 Preliminaries and Related Work

Text-Attributed Graphs. A text-attributed graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ denotes the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ the set of edges. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ encodes the graph structure, with $\mathbf{A}_{ij} = 1$ indicating the presence of an edge between nodes v_i and v_j. Each node v_i is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^F$, typically derived from unstructured text such as descriptions, documents, or user profiles. The node features for all nodes collectively form a feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$, which provides rich semantic information for downstream tasks.

LLMs for Graph Learning. Recent research has extensively explored applying LLMs to TAG-based tasks, yielding significant advancements in feature encoding, node classification, and link prediction [8, 9, 22, 47, 52, 56]. A predominant approach in this domain focuses on transforming graph structures and their associated textual attributes into carefully designed prompts, thereby enabling LLMs to leverage their language understanding and generation capabilities to serve as a direct predictor for TAG-related tasks [8, 9, 47].

In this paper, we focus on node classifiction which serves as the fundamental task of graph semi-supervised learning. LLM-based node classification is framed as a text-conditioned prediction problem. For each node $v_i \in V$, a prompt $\mathcal{P}(v_i)$ is constructed using its textual attribute $\mathbf{x}_i$, and optionally additional structural information such as neighbor features $\{\mathbf{x}_j : v_j \in \mathcal{N}(v_i)\}$. A language model $\mathcal{M}_\theta$ processes the prompt and outputs a predicted class label:

$$\hat{y}_i = \mathcal{M}_\theta(\mathcal{P}(v_i)),$$

where $\hat{y}_i \in \{1, \ldots, C\}$ and C is the number of classes. Depending on the setup, $\mathcal{M}_\theta$ may operate in zero-shot, few-shot, or fine-tuned mode using prompt-label pairs $\{(\mathcal{P}(v_i), y_i)\}_{v_i \in V_{\text{train}}}$. For instance, LLaGA [8] introduces a template-based methodology that encodes graph structures into textual prompt and subsequently trains a specialized projector for improved graph structure comprehension.

It is worth mentioning that while these approaches have demonstrated impressive performance by leveraging the language understanding capabilities of LLMs, they rely heavily on sufficient amounts of labeled data to perform instruction tuning on LLMs or train potential graph-text projectors for aligning between the graph and the semantic token space. Therefore, the application of LLMs for graph semi-supervised learning, where labeled data is scarce, remains largely unexplored despite its practical importance.

Pseudo Label Selection. In semi-supervised learning (SSL), pseudo-labeling [29, 43] serves as an effective solution by augmenting limited labeled datasets with generated labels for unlabeled data. To mitigate the potential label noise, previous research usually leverages model's confidence to select "easy" samples that are considered to be clean [28, 33]. Recent research, however, argues that there is little information to gain with these "easy" examples [36] and merely relying on high-confidence examples may make the model self-biased [31, 41]. Consequently, current research emphasizes the importance of mining both "easy" and "hard" samples during training to maximize model performance [31, 36, 44]. Nevertheless, identifying the "easiness" or "hardness" of samples is often non-trivial, especially for LLMs.

Large Language Models Preference Alignment. Preference alignment refers to the process of aligning the outputs of language models with human preferences, often focusing on safety, helpfulness, and factuality [2, 38]. Reinforcement Learning from Human Feedback (RLHF) [30, 46] is a prevalent method for aligning LMs with human preferences. The RLHF pipeline typically involves collecting human preference data, training a reward model, and using reinforcement learning to optimize the language model against this reward function [4, 11].

While RLHF has proven effective, its reliance on a separate reward model introduces computational and methodological complexities. To address these challenges, Direct Preference Optimization (DPO) [40] has emerged as a more efficient alternative, eliminating the need for an explicit reward model by directly optimizing preference probabilities. Several extensions and refinements of DPO have since been proposed. For instance, KTO [13] incorporates insights from prospect theory to enhance preference learning. Other variants, such as GPO [57], ΨPO [3], and ODPO [1], further improve or generalize DPO in different theoretical and practical aspects.

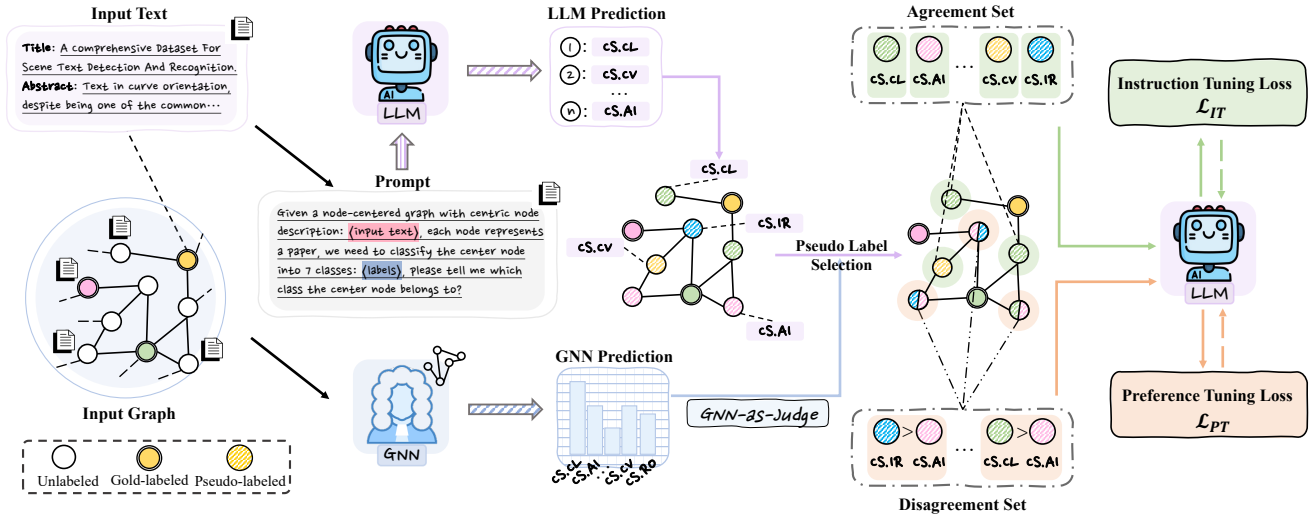


Figure 1: Framework of *GNN-as-Judge* for semi-supervised node classification on TAGs.

Recent advancements, including ORPO [19] and SimPO [34], simplify the alignment pipeline by removing the need for a reference model while maintaining competitive performance. Despite these advancements, the application and adaptation of preference alignment techniques to structured tasks such as graph-based learning remain underexplored.

3 Proposed Approach

In this section, we propose an LLM-GNN pseudo co-labeling framework *GNN-as-Judge* that addresses LLM pseudo-labeling challenges through two core designs as shown in Figure 1: (i) a collaborative pseudo label selection mechanism that leverages GNN as complementary signal sources for LLM to generate high-quality pseudo labels, and (ii) a weakly-supervised fine-tuning algorithm that mitigates label noise when fine-tuning LLM with generated pseudo labels.

3.1 GNN-as-Judge for LLM Pseudo Labeling on Graphs

To break the bottleneck of using LLM-generated pseudo labels, our approach tries to mine "easy" and "hard" pseudo labels based on the agreement and disagreement between LLM and GNN, in order to provide reliable pseudo labels for fine-tuning the LLMs. We use complementary strengths of two distinct models: a structure-aware GNN $\mathcal{M}_{\text{GNN}}$ and a text-centric LLM $\mathcal{M}_{\text{LLM}}$, both trained on the labeled set V_L. For each unlabeled node $v_i \in V_U$, we generate two pseudo labels using both models: $\hat{y}_i^{\text{GNN}} = \arg\max_j M_{\text{GNN}}^{\theta}(x_i)_j$ from the GNN, and $\hat{y}_i^{\text{LLM}} = \texttt{Parse}(\mathcal{M}_{\text{LLM}}(x_i))$ from the LLM. Based on prediction agreement, we partition the unlabeled set into two disjoint subsets: the *agreement node set* $\mathcal{V}_{\text{agreed}} = \{v_i \in V_U \mid \hat{y}_i^{\text{GNN}} = \hat{y}_i^{\text{LLM}}\}$ contains "easy" samples and the *disagreement node set* $\mathcal{V}_{\text{disagreed}} = \{v_i \in V_U \mid \hat{y}_i^{\text{GNN}} \neq \hat{y}_i^{\text{LLM}}\}$ represents "hard" samples. To improve the reliability of the pseudo-labeled nodes, we further refine those two node sets based on GNN feedback as follows.

Agreement Node Set Selection with GNN Feedback. While we assume that nodes in agreement set are more likely to have correct pseudo labels as agreement between models with distinct inductive biases suggests higher label reliability, potential noisy labels may still exist within the agreement node set and need further refinement. In semi-supervised learning (SSL), prediction confidence is often used as an indicator of a model's certainty in its own outputs [29, 43]. However, directly estimating the confidence of LLM predictions is computationally expensive and prone to be self-biased [45, 53]. To address these challenges, we treat the GNN as an pseudo label judge and further refine the agreement node set $\mathcal{V}_{\text{agreed}}$ into a high-quality, diverse subset $\mathcal{V}'_{\text{agreed}}$ based on GNN confidence.

Class-dependent Confidence-Based Selection. We first apply a confidence-based filtering to eliminate potentially unreliable pseudo-labels. For each node $v_i \in \mathcal{V}_{\text{agreed}}$, we quantify the GNN's certainty using its maximum softmax probability assigned to the agreed-upon class $c = \hat{y}_i^{\text{GNN}}$:

$$C_{\text{GNN}}(v_i) = P_{\text{GNN}}(c|v_i) = \max_{j \in \{1,\ldots,K\}} P_{\text{GNN}}(j|v_i), \tag{1}$$

where $P_{\text{GNN}}(j|v_i)$ denotes the probability assigned by the GNN to class j for node v_i. We define a confidence threshold τ_{conf} and retain only the nodes satisfying $C_{\text{GNN}}(v_i) \geq \tau_{\text{conf}}$, forming a filtered set: $\mathcal{V}_{\text{filtered}} = \{v_i \in \mathcal{V}_{\text{agreed}} \mid C_{\text{GNN}}(v_i) \geq \tau_{\text{conf}}\}$. This ensures that all candidate pseudo-labels meet a minimum reliability standard according to the GNN.

Although confidence filtering improves overall label quality, it may inadvertently create class imbalance by excluding more samples from classes that inherently yield lower confidence scores. To preserve class diversity and avoid selection bias toward classes with inherently higher confidence values, we further implement class-aware selection within each class. Specifically, we partition the

filtered set by predicted class as $R'_c = \{v_i \in \mathcal{V}_{\text{filtered}} \mid \hat{y}_i^{\text{GNN}} = c\}$, and within each R'_c, we select the top β percent of nodes ranked by descending GNN confidence, where the number of selected nodes is $k'_c = \lceil \beta \cdot |R'_c| \rceil$, for each $c \in \{1, \ldots, K\}$, and β is a predefined hyperparameter controlling the selection ratio. The final pseudo-labeled subset is then given by:

$$\mathcal{V}'_{\text{agreed}} = \left\{ v_i \in \mathcal{V}_{\text{filtered}} \;\middle|\; \text{Rank}_{C_{\text{GNN}}}(v_i) \leq \beta \cdot |R'_c|, \hat{y}_i^{\text{GNN}} = c \right\} \tag{2}$$

where $\text{Rank}_{C_{\text{GNN}}}(v_i)$ denotes the ranking position of node v_i within class c based on descending $C_{\text{GNN}}(v_i)$.

Disagreement Node Set Selection with GNN Feedback. LLMs and GNNs exhibit fundamentally different inductive biases: LLMs are designed for extracting rich semantic information from raw text, while GNNs are strong at capturing structural knowledge through message-passing. Since our goal is to fine-tune the LLM to improve its performance, simply using easy, self-generated labels where the LLM already predicts correctly provides limited new learning signals and may lead to overfitting. Instead, we further utilize the disagreement node set $\mathcal{V}_{\text{disagreed}}$, where the LLM is more likely to produce incorrect pseudo labels due to its lack of structural awareness.

Preference Assessment with GNN Feedback. GNN's probability distribution over different classes provides a natural indicator of its preference strength. For each node $v_i \in \mathcal{V}_{\text{disagreed}}$, we compute a preference score measuring how strongly the GNN favors its own prediction over the LLM's prediction: $S_{\text{pref}}(v_i) = P_{\text{GNN}}(\hat{y}_i^{\text{GNN}} \mid v_i) - P_{\text{GNN}}(\hat{y}_i^{\text{LLM}} \mid v_i)$, where $P_{\text{GNN}}(\hat{y}_i^{\text{GNN}} \mid v_i)$ is the GNN's predicted probability for its top class, and $P_{\text{GNN}}(\hat{y}_i^{\text{LLM}} \mid v_i)$ is the probability assigned to the LLM's predicted class. A larger preference score indicates a stronger conviction by the GNN in its own prediction relative to the LLM's alternative.

Structural Importance Refinement. We also want to prioritize nodes with rich structural information. GNNs rely on message-passing mechanisms that aggregate information from neighbors to learn representative embeddings [27, 54]. Due to the power-law distribution of node degrees in many real-world graphs, low-degree nodes receive less information during aggregation, potentially leading to higher error rates [48]. Inspired by this idea, we hypothesize that LLMs – lacking access to graph topology – are more prone to errors on nodes rich in structural information, whereas GNNs can leverage such connections for improved predictions. We further validate this hypothesis through a detailed case study presented in Appendix F, where we analyze the correlation between node structural properties and the respective performances of LLMs and GNNs. To capture the structural importance of nodes, we compute the PageRank score [39] $S_{\text{PR}}(v_i)$:

$$S_{\text{PR}}(v_i) = (1-\alpha) \cdot \frac{1}{|V|} + \alpha \sum_{v_j \in N_{in}(v_i)} \frac{S_{\text{PR}}(v_j)}{|N_{out}(v_j)|}, \tag{3}$$

where α is a damping factor, $|V|$ is the total number of nodes, $N_{in}(v_i)$ is the set of nodes with edges pointing to v_i, and $|N_{out}(v_j)|$ is the out-degree of node v_j. This recursive definition quantifies each node's centrality and influence, with higher scores indicating nodes that leverage more extensive graph connections.

To create a balanced selection that accounts for both preference strength and structural importance, we convert each metric to percentile ranks and combine them as $S_{\text{combined}}(v_i) = \beta_1 \cdot r_{\text{pref}}(v_i) + \beta_2 \cdot r_{\text{PR}}(v_i)$, where $r(v_i)$ denotes the high-to-low ranking percentage, and β_1 and β_2 control the relative importance of preference signals and structural information, respectively. We then select the final disagreement nodes based on this combined score:

$$\mathcal{V}'_{\text{disagreed}} = \{v_i \in \mathcal{V}_{\text{disagreed}} \mid \text{Rank}_{S_{\text{combined}}}(v_i) \leq \gamma \cdot |\mathcal{V}_{\text{disagreed}}|\}, \tag{4}$$

where γ is a hyperparameter controlling the proportion of nodes to select from the disagreement set.

> ***Remark***: While existing works in SSL emphasize the importance of balancing between both "easy" and "hard' samples [5, 28, 36], directly utilizing LLMs to evaluate "easiness" or "hardness" of unlabeled nodes on TAGs remains non-trivial. Our strategy leverages the agreement between LLM and GNN predictions, using the GNN as a pseudo label judge to effectively identify "easy" and "hard" samples, providing a practical method to identify reliable pseudo labels for fine-tuning LLMs for semi-supervised node classification.

3.2 LLM Weakly-Supervised Fine-Tuning on Graphs with GNN Feedback

Based on our selected pseudo-labeled nodes, we propose a weakly-supervised fine-tuning algorithm that fine-tunes LLMs on graphs using a unified objective. Our approach integrates both instruction tuning and preference tuning into a single training framework:

$$\begin{aligned} \mathcal{L}(\theta) = \; & \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}_{\text{agreed}'}} [\mathcal{L}_{\text{IT}}(\theta; x_i, y_i)] \\ & + \lambda \mathbb{E}_{(x_i, y_{w,i}, y_{l,i}) \sim \mathcal{D}_{\text{disagreed}'}} [\mathcal{L}_{\text{PT}}(\theta; x_i, y_{w,i}, y_{l,i})] \end{aligned} \tag{5}$$

where $\mathcal{D}_{\text{agreed}'}$ represents the data distribution over the selected agreement node set $\mathcal{V}'_{\text{agreed}}$, $\mathcal{D}_{\text{disagreed}'}$ represents the data distribution over the selected disagreement node set $\mathcal{V}'_{\text{disagreed}}$, and λ controls the contribution of the preference tuning loss.

LLM Instruction Tuning with LLM-GNN Agreement. For nodes in the agreement set $\mathcal{V}'_{\text{agreed}}$, we apply instruction tuning [38] to reinforce correct predictions. Given an input node text feature x_i and the selected agreed pseudo label y_i, the instruction tuning loss is defined as:

$$\mathcal{L}_{\text{IT}}(\theta; x_i, y_i) = -\log p_\theta(y_i | x_i), \tag{6}$$

where $p_\theta(y_i|x_i)$ represents the LLM's probability of generating the label y_i given the node text x_i, and θ denotes the model parameters. This maximum likelihood objective reinforces patterns where both the LLM and GNN already exhibit strong performance, consolidating the model's understanding of straightforward cases.

LLM Preference Tuning with LLM-GNN Disagreement. For the selected disagreed nodes $\mathcal{V}'_{\text{disagreed}}$, where GNN and LLM predictions differ, standard instruction tuning with potentially noisy pseudo labels is suboptimal. Even though we leverage the GNN's

prediction as the preferred response, forcing the LLM to strictly match a potentially incorrect or biased GNN prediction through instruction tuning could lead to potentially compromising its original performance and inherent text understanding capabilities. Instead, we leverage these informative disagreement cases through preference tuning. Preference tuning optimizes LLM responses based on paired examples with a preferred and non-preferred output [3, 11]. Unlike instruction tuning, which forces models to predict an absolute label, preference tuning enables the model to learn from the relative relationship between competing outputs. The general preference tuning objective can be formulated as:

$$\mathcal{L}_{\text{PT}}(\theta; x_i, y_{w,i}, y_{l,i}) = -\log \sigma(f_\theta(x_i, y_{w,i}, y_{l,i})) \tag{7}$$

where x_i represents the input prompt for node v_i, $y_{w,i}$ and $y_{l,i}$ are the preferred and non-preferred outputs respectively, and $f_\theta(\cdot)$ is a preference function that scores the relative preference between the two outputs. In our implementation, we adopt *Odds Ratio Preference Optimization (ORPO)* [19], which uses the log odds ratio as the preference function:

$$f_\theta(x, y_w, y_l) = \log \frac{\text{odds}_\theta(y_w \mid x)}{\text{odds}_\theta(y_l \mid x)} \tag{8}$$

where $\text{odds}_\theta(y \mid x) = P_\theta(y \mid x)/(1 - P_\theta(y \mid x))$. By minimizing this loss over the set $\mathcal{V}'_{\text{disagreed}}$, the LLM learns to increase the relative likelihood of the GNN's predictions over LLM's predictions, mitigating the potential risk of overfitting to noisy pseudo labels.

Remark: Our framework can be considered as an LLM preference alignment framework by replacing human feedback with signals derived from GNNs. The agreement node set $\mathcal{V}'_{\text{agreed}}$ provides supervised training data, while the disagreement set $\mathcal{V}'_{\text{disagreed}}$ is transformed into preference pairs with GNN outputs as preferred responses. While we implement preference tuning using ORPO [19], our approach is also compatible with other methods like such as DPO [40], SimPO [34], and other variants.

4 Experiments

We conduct comprehensive experiments to validate the effectiveness of our framework. These experiments aim to investigate the following research questions:

- **RQ1:** How does our framework perform in comparison to baseline models in node classification under various label rate conditions across datasets?
- **RQ2:** How effectively does the model adapt to unseen datasets in zero-shot settings?
- **RQ3:** What is the core contribution of our `GNN-as-Judge`, specifically the pseudo-label selection strategy and weakly-supervised fine-tuning pipeline, to the overall performance?
- **RQ4:** How do different hyperparameter choices and variants of preference tuning losses affect our framework's performance?

4.1 Experimental Setup

Datasets.

We train and evaluate our framework on four widely-used benchmark datasets for semi-supervised node classification: Cora [55], Citeseer [42], Pubmed [42] and ArXiv [37]. For Cora, Citeseer and Pubmed, which are three most widely used citation networks, we follow the experimental setup of previous work [15] and split each dataset into training (i.e., K nodes per class for K-shot task), validation set and test set. To evaluate performance on large-scale graphs, we also include the ArXiv dataset from the Open Graph Benchmark (OGB) [21]. For ogbn-arxiv, we randomly sample different percentages (1%, 5%, and 10%) of nodes from the training split as labeled data, while maintaining the standard validation and test splits from the original benchmark. Statistics for these datasets are summarized in Table 1. More detailed statistics of all datasets are summarized in Appendix A.

Table 1: Summary statistics of the evaluation datasets.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
ogbn-arxiv	169,343	1,166,243	128	40

Baselines. To evaluate the effectiveness of our proposed framework, we compare it against baseline methods from two primary categories: (1) *GNN-as-Predictors*: We include established graph neural network models such as GCN [27], GraphSAGE [18], GAT [49], and SGC [51]; (2) *LLM-as-Predictors*: Aligning with our focus, we benchmark against various LLM-based methods. This includes base LLMs with different prompting strategies, such as zero-shot, chain-of-thought, and neighbor-augmented prompting, as well as graph-focused LLM fine-tuning methods like instruction tuning, GraphGPT [47], and LLaGA [8]. Further details on these baselines are provided in the Appendix B.

Implementation Details. For our experiments, we maintain consistent data splits across all evaluated models to ensure fair comparison. Performance is measured using accuracy, with results reported as mean and standard deviation over five independent runs. For the GNN component of our framework, we implement a 2-layer GCN architecture [27] with 64-dimensional hidden representations. The LLM components in our *GNN-as-Judge* framework utilize Llama-3-8B-Instruct [16] and Mistral-7B-Instruct-v0.2 [25] as base models, allowing us to evaluate performance across different model families. We fine-tune these models using LoRA [20] with rank 8 and $\alpha = 16$ to efficiently adapt the pre-trained LLMs while maintaining computational feasibility. For experiments other than overall performance comparisons, we employ Llama-3-8B-Instruct as our primary model due to its stronger performance in initial tests. Additional training details, including hyperparameters and optimization settings, are provided in Appendix C.

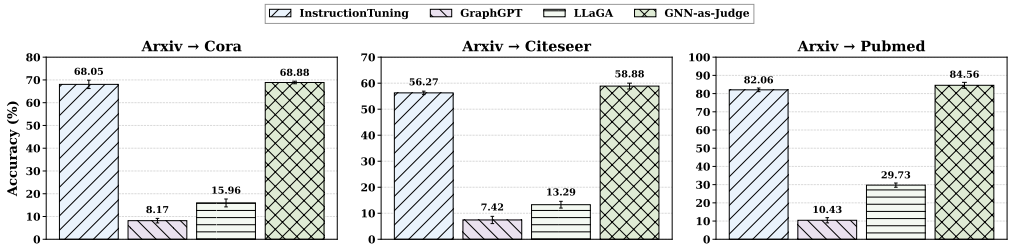
4.2 Overall Performance Comparison (RQ1)

Tables 2 and 3 present the comparative performance of our approach against various baseline methods across multiple datasets and experimental settings. From these comprehensive results, we observe several key findings:

Table 2: Averaged node classification accuracy (%) on the `Cora`, `Citeseer`, and `Pubmed` datasets with 5-shot and 10-shot learning. Values shown are mean ± standard deviation. Best results are bolded, while second-best results are underlined.

	Methods	Cora		Citeseer		Pubmed	
		5-shot	10-shot	5-shot	10-shot	5-shot	10-shot
N/A	GCN	71.72±0.22	78.22±0.89	63.24±0.87	68.38±1.49	70.58±0.49	75.33±0.94
	GraphSAGE	70.66±1.09	77.98±0.32	62.35±0.83	67.02±0.56	68.59±1.01	75.12±0.55
	SGC	71.48±0.35	78.44±0.37	62.08±0.77	67.44±0.60	70.74±0.94	74.98±1.91
	GAT	71.70±0.57	77.58±0.23	63.48±1.54	68.10±1.77	69.56±1.55	72.84±1.36
`Mistral`	Zero-Shot	59.78±1.26	59.78±1.26	42.56±0.33	42.56±0.33	77.24±1.15	77.24±1.15
	Graph-CoT	58.34±0.72	58.34±0.72	37.94±2.13	37.94±2.13	82.06±2.77	82.06±2.77
	w. Neighbor	66.69±0.43	66.69±0.43	59.65±1.23	59.65±1.23	72.49±1.82	72.49±1.82
	InstructTuning	64.14±1.22	65.77±0.56	61.05±0.36	68.12±0.73	85.33±0.46	84.91±1.04
	LLaGA	58.93±0.72	71.17±0.25	46.78±1.02	49.96±0.64	52.03±2.35	65.23±2.56
	GraphGPT	52.35±1.74	59.66±0.42	43.08±0.94	46.62±0.58	52.46±2.19	62.18±2.31
	GNN-as-Judge	76.12±0.73	**79.72±1.03**	71.04±0.35	73.30±0.57	86.36±0.82	86.44±0.93
`Llama 3`	Zero-Shot	65.54±0.26	65.54±0.26	58.17±0.64	58.17±0.64	74.51±0.39	74.51±0.39
	Graph-CoT	63.02±0.77	63.02±0.77	47.23±1.06	47.23±1.06	86.22±2.47	86.22±2.47
	w. Neighbor	68.72±1.56	68.72±1.56	54.93±1.28	54.93±1.28	74.98±3.16	74.98±3.16
	InstructTuning	69.33±1.50	69.53±1.21	68.77±2.66	69.10±1.63	83.71±0.64	85.03±0.16
	LLaGA	62.88±2.19	69.25±0.97	43.71±4.36	51.22±1.43	58.63±1.05	67.29±2.26
	GraphGPT	60.17±1.44	61.58±0.77	51.83±2.24	55.40±3.16	57.39±3.67	71.33±2.81
	GNN-as-Judge	**77.35±1.13**	78.33±0.86	**71.54±1.10**	**73.33±0.38**	**90.45±0.78**	**91.01±0.47**

Figure 2: Zero-shot cross-dataset node classification results. `Llama-3-8B` trained on `ArXiv` with 1% label rate were evaluated on `Cora`, `Citeseer`, and `PubMed` without fine-tuning.



- Our *GNN-as-Judge* consistently outperforms both traditional GNN architectures and existing LLM-based methods across all datasets, demonstrating the effectiveness of our collaborative pseudo label selection strategy and proposed weakly-supervised fine-tuning approach. The improvements are systematic rather than dataset-specific, indicating the robustness of our framework.
- The performance advantages of our method are particularly pronounced in low-label ratio settings where labeled data is extremely scarce. In the challenging 5-shot scenario, our approach shows significant improvements over competitive baselines across all datasets, with gains ranging from 5-8% over the strongest baselines. Specifically, on Cora, our approach achieves 77.35% accuracy with `Llama 3` in the 5-shot setting, surpassing the best GNN baseline (GCN) by 5.63% and the best LLM baseline (InstructTuning) by 8.02%.
- While traditional GNNs maintain relatively stable performance as label ratios decrease due to their structural inductive bias, most LLM-based approaches show substantial degradation when transitioning from higher to lower resource settings. More concerning, specialized graph-aware LLM methods such as LLaGA and GraphGPT, which utilize graph tokens or structured representations, often perform worse than their zero-shot counterparts in extreme low-resource settings.
- Our method's consistent superior performance across different LLM families – `Mistral-7B` and `Llama-3-8B` – confirms that our approach achieves excellent results regardless of the underlying LLM architecture. This model-agnostic effectiveness makes our framework a versatile solution applicable to various LLM backbones, suggesting that the benefits stem from our propose *GNN-as-Judge* pipeline rather than model-specific optimizations.
- The performance gap between our method and baselines tends to be larger on datasets with more complex structural patterns (Citeseer, ArXiv) compared to those with clearer community structures (Cora), suggesting that our GNN-as-Judge framework is particularly effective at leveraging structural information in challenging graph learning scenarios where semantic and structural cues must be carefully balanced.

Table 3: Node classification accuracy (%) on the ArXiv dataset with varying label rates, and on Cora, Citeseer, and Pubmed with 20-shot. Values shown are mean ± standard deviation. Best results are bolded, while second-best results are underlined.

	Methods	ArXiv			Others (20-shot)		
		1%	2.5%	5%	Cora	Citeseer	Pubmed
N/A	GCN	59.75±1.28	62.36±0.78	66.17±0.34	81.30±0.59	69.75±0.42	79.64±0.98
	GraphSAGE	59.93±0.89	61.42±0.62	64.29±0.41	81.37±0.43	69.95±0.66	77.64±0.92
	SGC	55.72±0.55	59.86±0.47	62.93±0.38	79.84±0.68	70.92±0.77	78.45±0.56
	GAT	59.48±1.86	63.05±0.91	66.21±0.34	80.94±0.56	70.02±0.44	78.37±0.38
Mistral	Zero-Shot	43.06±2.45	43.06±2.45	43.06±2.45	59.78±1.26	42.56±0.33	77.24±1.15
	Graph-CoT	53.89±3.18	53.89±3.18	53.89±3.18	58.34±0.72	37.94±2.13	82.06±2.77
	w. Neighbor	42.57±1.38	42.57±1.38	42.57±1.38	66.69±0.43	59.65±1.23	72.49±1.82
	InstructTuning	53.43±1.02	59.32±0.84	64.83±0.36	70.02±1.56	66.77±1.56	86.64±1.32
	LLaGA	59.55±1.37	67.79±1.49	69.95±1.77	77.91±1.12	63.57±0.56	68.37±2.05
	GraphGPT	65.67±1.03	66.35±1.64	68.06±2.45	64.16±0.98	65.27±1.39	79.91±1.47
	GNN-as-Judge	64.72±0.74	66.93±0.82	69.78±1.24	81.28±1.94	**74.05±0.47**	90.87±2.16
Llama 3	Zero-Shot	50.18±1.44	50.18±1.44	50.18±1.44	65.54±0.26	58.17±0.64	74.51±0.39
	Graph-CoT	52.49±1.32	52.49±1.32	52.49±1.32	63.02±0.77	47.23±1.06	86.22±2.47
	w. Neighbor	49.28±2.09	49.28±2.09	49.28±2.09	68.72±1.56	54.93±1.28	74.98±3.16
	InstructTuning	57.32±0.83	60.54±0.95	66.76±0.79	71.89±1.76	68.44±0.93	88.16±2.17
	LLaGA	57.65±0.82	65.19±1.08	67.63±0.44	76.51±0.22	66.17±0.43	68.77±0.35
	GraphGPT	64.38±0.75	67.68±0.45	69.56±0.77	63.25±0.34	62.84±0.49	78.36±1.24
	GNN-as-Judge	**67.57±0.94**	**68.42±0.65**	**70.28±0.81**	**81.96±1.43**	73.79±1.69	**91.04±0.77**

4.3 Cross-Dataset Zero-Shot Node Classification (RQ2)

In this section, we evaluate the zero-shot generalization capabilities of *GNN-as-Judge*. We trained various LLM-based graph learning models on the ArXiv dataset (1% label rate) and evaluated their zero-shot performance on Cora, Citeseer, and Pubmed without additional fine-tuning. Unlike traditional GNNs that require task-specific classification heads, LLM-based methods can perform zero-shot learning across different label sets. As shown in Figure 2, *GNN-as-Judge* demonstrates superior zero-shot transfer performance across all target datasets. It substantially outperforms GraphGPT and LLaGA, which struggle with cross-dataset generalization. Their approach of encoding graph structure into tokens appears to constrain the LLM's generalization capabilities, resulting in performance worse than untuned base LLMs. These results indicate that GNN-as-Judge is more robust to distribution shifts between graph datasets and better preserves the LLM's inherent generalization capabilities while incorporating graph-based insights. This makes our approach particularly valuable for practical applications where labeled data may be scarce or available only for specific domains, necessitating models that can effectively generalize to new, unseen graph structures.

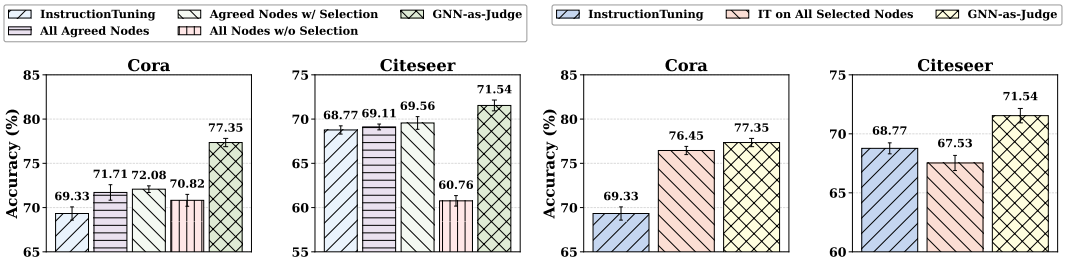
4.4 Ablation Study (RQ3)

In this section, we conduct an ablation study to analyze the contribution of our proposed *GNN-as-Judge*, specifically the pseudo label selection strategy and weakly-supervised fine-tuning pipeline to the overall performance. In our experiments, *InstructionTuning* refers to the standard instruction tuning using ground truth labels, while *All Agreed Nodes* trains on all nodes where GNN and LLM predictions match. *Agreed Nodes w/ Selection* utilizes nodes selected by our proposed selection strategy, and *All w/o Selection* uses all nodes without any selection criteria.

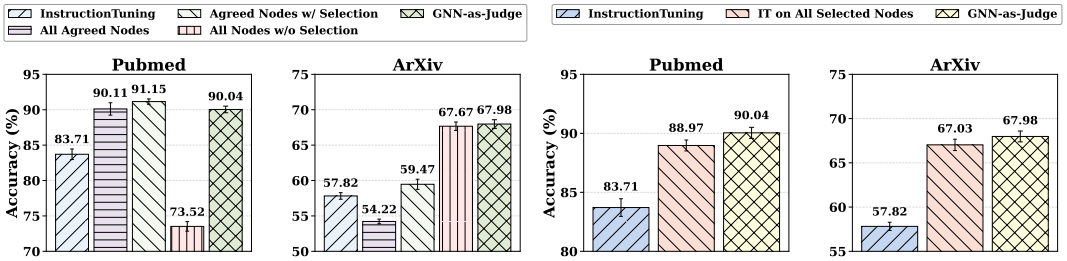
Effectiveness of Pseudo Label Selection. As shown in left panels of Figure 3 (a) and (b), training on all nodes without selection leads to degraded performance across all four datasets, with particularly notable drops on Citeseer and ArXiv. This observation strongly underscores the crucial problem of label noise when attempting to train using pseudo labels derived from the entire graph. Conversely, focusing instruction tuning on a carefully selected subset of agreed nodes consistently improves performance over training on all nodes across Cora, Citeseer, Pubmed, and ArXiv datasets, demonstrating the fundamental benefit of selection. Furthermore, our proposed *GNN-as-Judge* selection strategy yields superior performance compared to methods that only use selected agreed nodes on all four datasets. This indicates that the *GNN-as-Judge* is effective not just at finding easy training samples, but crucially, it leverages the structural information from the GNN signals to identify a higher-quality, more informative set of training nodes across diverse graph domains and scales.

Effectiveness of Weakly-Supervised Fine-Tuning. The right panels of Figure 3 (a) and (b) present results comparing different fine-tuning strategies applied after node selection. Our full GNN-as-Judge approach consistently outperforms standard instruction tuning on selected nodes (*IT on All Selected Nodes*) across all datasets, with particularly notable gains on Citeseer due to potentially more label noise in the disagreement set. This demonstrates the effectiveness of our specifically designed weakly-supervised fine-tuning strategy that combines instruction tuning with preference learning to distill knowledge while mitigating noise. Unlike standard instruction tuning on pseudo-labeled data, our approach is better equipped to handle the inherent uncertainty or potential noise present in the pseudo labels of the selected nodes.

Figure 3: Ablation studies: (a) results on Cora and Citeseer datasets; (b) results on Pubmed and ArXiv datasets.



(a) Left: Ablation study showing the effectiveness of our proposed pseudo label selection approach on Cora and Citeseer. Right: Ablation study showing the effectiveness of our proposed weakly-supervised fine-tuning algorithm on Cora and Citeseer.

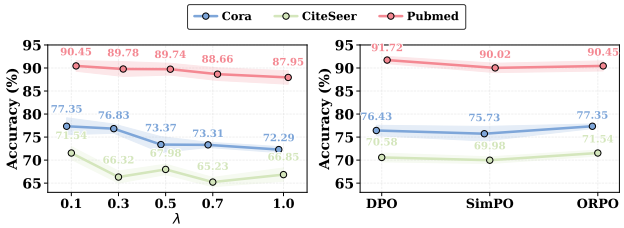


(b) Left: Ablation study showing the effectiveness of our proposed pseudo label selection approach on Pubmed and ArXiv. Right: Ablation study showing the effectiveness of our proposed weakly-supervised fine-tuning algorithm on Pubmed and ArXiv.

4.5 Analysis of Preference Tuning Variants and Parameter Sensitivity (RQ4)

In this section, we further analyze hyperparameter choices for proposed weakly-supervised fine-tuning approach and evaluate the framework's robustness to different preference optimization algorithms.

Figure 4: Left: Sensitivity to the λ parameter controlling the balance between instruction tuning and preference tuning. Right: Comparison of different preference tuning losses.



Sensitivity to λ. In our approach, the hyperparameter λ controls the balance between standard instruction tuning and preference tuning as shown in Eq. 5. Figure 4 (left) illustrates how different λ values affect performance on Cora, Citeseer and Pubmed datasets in 5-shot settings. We observe a clear trend where lower λ values yield significantly better results on both datasets, with performance declining as λ increases. This suggests that using a moderate weight for preference tuning loss will maintain better model performance.

Adaptation to Different Preference Tuning Losses. We evaluate the performance of our *GNN-as-Judge* framework across different preference tuning losses, comparing DPO [40], SimPO [34], and ORPO [19] on Cora, CiteSeer and Pubmed datasets in 5-shot setting. As shown in Figure 4 (right), these preference tuning methods demonstrate relatively comparable performance, with ORPO showing slightly higher accuracy on both Cora and Citeseer datasets. Notably, all variants of our framework incorporating various preference tuning losses consistently outperform performing standard instruction tuning on all selected data, with particularly pronounced improvements on the CiteSeer dataset where noisy labels are more prevalent. This observation highlights that our *GNN-as-Judge* is a plug-and-play framework where different preference tuning losses can be seamlessly integrated without compromising overall performance.

5 Conclusion

In this paper, we present *GNN-as-Judge*, a novel framework that addresses the challenge of applying LLMs to semi-supervised graph learning with limited labeled data. Our approach leverages complementary strengths from both GNNs and LLMs through two key mechanisms: (i) a strategic pseudo-label selection strategy that identifies both high-confidence agreement and structurally informative disagreement nodes, and (ii) a weakly-supervised fine-tuning algorithm combining instruction tuning with preference tuning to mitigate label noise.

References

[1] Afra Amini, Tim Vieira, and Ryan Cotterell. 2024. Direct preference optimization with an offset. *arXiv preprint arXiv:2402.10571* (2024).

[2] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861* (2021).

[3] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *AISTATS*.

[4] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862* (2022).

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

[7] Jonathan Chang and David Blei. 2009. Relational Topic Models for Document Networks. In *AISTATS*.

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

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

[10] Hao Cheng, Zhaowei Zhu, Xingyu Li, Yifei Gong, Xing Sun, and Yang Liu. 2020. Learning with instance-dependent label noise: A sample sieve approach. In *NeurIPS*.

[11] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *NeurIPS*.

[12] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234* (2022).

[13] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. In *ICML*.

[14] Yang Gao, Dana Alon, and Donald Metzler. 2024. Impact of preference noise on the alignment performance of generative language models. In *COLM*.

[15] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).

[16] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[17] Jiayan Guo, Lun Du, Hengyu Liu, Mengyu Zhou, Xinyi He, and Shi Han. 2023. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066* (2023).

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

[19] Jiwoo Hong, Noah Lee, and James Thorne. 2024. ORPO: Monolithic Preference Optimization without Reference Model. In *EMNLP*.

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models.. In *ICLR*.

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

[22] Zhengyu Hu, Yichuan Li, Zhengyu Chen, Jingang Wang, Han Liu, Kyumin Lee, and Kaize Ding. 2024. Let's Ask GNN: Empowering Large Language Model for Graph In-Context Learning. In *EMNLP Findings*.

[23] Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2023. Can LLMs Effectively Leverage Graph Structural Information through Prompts, and Why? *Trans. Mach. Learn. Res.* (2023).

[24] Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2024. Can LLMs Effectively Leverage Graph Structural Information through Prompts, and Why? *TMLR* (2024).

[25] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. *https://arxiv.org/abs/2310.06825* (2023).

[26] Taehyeon Kim, Joonkee Kim, Gihun Lee, and Se-Young Yun. 2023. Instructive decoding: Instruction-tuned large language models are self-refiner from noisy instructions. In *ICLR*.

[27] Thomas N Kipf and Max Welling. 201T. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[28] M Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *NeurIPS*.

[29] Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*.

[30] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. 2018. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871* (2018).

[31] Hongrui Liu, Binbin Hu, Xiao Wang, Chuan Shi, Zhiqiang Zhang, and Jun Zhou. 2022. Confidence may cheat: Self-training on graph neural networks under distribution shift. In *WWW*.

[32] Junyu Luo, Xiao Luo, Kaize Ding, Jingyang Yuan, Zhiping Xiao, and Ming Zhang. 2024. RobustFT: Robust Supervised Fine-tuning for Large Language Models under Noisy Response. *arXiv preprint arXiv:2412.14922* (2024).

[33] Fan Ma, Deyu Meng, Qi Xie, Zina Li, and Xuanyi Dong. 2017. Self-paced co-training. In *ICML*.

[34] Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. In *NeurIPS*.

[35] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196* (2024).

[36] Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Uncertainty-aware self-training for few-shot text classification. In *NeurIPS*.

[37] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. 2012. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*.

[38] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*.

[39] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford infolab.

[40] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*.

[41] Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. 2021. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. In *NeurIPS*.

[42] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* (2008).

[43] Weiwei Shi, Yihong Gong, Chris Ding, Zhiheng MaXiaoyu Tao, and Nanning Zheng. 2018. Transductive semi-supervised deep learning using min-max features. In *ECCV*.

[44] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. 2016. Training region-based object detectors with online hard example mining. In *CVPR*.

[45] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. 2023. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493* (2023).

[46] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *NeurIPS*.

[47] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *SIGIR*.

[48] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Investigating and mitigating degree-related biases in graph convoltuional networks. In *ACM International Conference on Information & Knowledge Management*.

[49] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[50] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

[51] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *ICML*.

[52] Xixi Wu, Yifei Shen, Fangzhou Ge, Caihua Shan, Yizhu Jiao, Xiangguo Sun, and Hong Cheng. 2025. A Comprehensive Analysis on LLM-based Node Classification Algorithms. *arXiv preprint arXiv:2502.00829* (2025).

[53] Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. 2024. Pride and Prejudice: LLM Amplifies Self-Bias in Self-Refinement. In *ACL*.

[54] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. 2021. Graphformers: Gnn-nested transformers for representation learning on textual graph. In *NeurIPS*.

[55] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*.
[56] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *EACL*.
[57] Siyan Zhao, John Dang, and Aditya Grover. 2024. Group preference optimization: Few-shot alignment of large language models. In *ICLR*.

A Dataset Statistics

Table 4: Summary statistics of the evaluation datasets.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
ogbn-arxiv	169,343	1,166,243	128	40

Table 4 presents the detailed statistics of the datasets used in our experiments, including the number of nodes, edges, features, and classes for each dataset. These datasets all consist of nodes representing academic papers, with edges indicating citation relationships. The associated text attributes include each paper's title and abstract.

Within each dataset, nodes are labeled according to their academic category, as detailed in Table 5. For example, the arXiv dataset includes 40 computer science sub-categories such as cs.AI (Artificial Intelligence) and cs.DB (Databases).

B Baseline Methods

To evaluate the effectiveness of our proposed framework, we compare it against established methods from two primary categories: GNN-as-Predictors and LLM-as-Predictors.

B.1 GNN-as-Predictors Methods

In this category, we include four classic graph neural network architectures:

- **GCN** [27]: Graph Convolutional Networks perform neighborhood aggregation through spectral convolutions, using a simple and effective message-passing scheme.
- **GraphSAGE** [18]: This inductive framework samples and aggregates features from a node's local neighborhood, enabling generalization to previously unseen nodes.
- **GAT** [49]: Graph Attention Networks leverage masked self-attentional layers to weight neighbor importance, allowing the model to focus on the most relevant connections.
- **SGC** [51]: Simple Graph Convolution simplifies GCNs by removing nonlinearities between layers and collapsing weight matrices, resulting in a single linear transformation followed by a softmax classifier, while maintaining competitive performance with significantly reduced computational complexity.

These models serve as strong baselines that rely primarily on graph structure and node features without leveraging the reasoning capabilities of language models.

B.2 LLM-as-Predictors Methods

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. We evaluate both fine-tuned and zero-shot LLM approaches:

Fine-tuned LLM Methods:

- **LLaGA** [8]: Employs a multi-step approach where node text is first encoded via a language model, then processed through a GNN, concatenated across layers, projected into the LLM's dimensionality, and finally combined with instructions for label prediction. Only the projection layer parameters are tuned using next-token-prediction loss.
- **GraphGPT** [47]: Implements a more complex framework with three distinct pre-training and instruction tuning stages to effectively integrate graph structure with language understanding.

Zero-shot LLM Methods:

- **Chain-of-Thought** [50]: Enables step-by-step reasoning by breaking down complex graph-related problems into sequential reasoning steps.
- **Neighbor-Augmented Prompting** [23]: Enriches prompts with structural information from node neighborhoods, providing context about graph topology to enhance zero-shot performance.

These LLM-based methods represent the state-of-the-art in leveraging large language models for graph-related tasks, with varying approaches to integrating structural information and text understanding.

C Implementation Details

This section provides comprehensive details about the implementation, hyperparameter settings, and training procedures used in our experiments.

Data Splits. For the Cora, Citeseer, and Pubmed datasets, we follow standard node classification protocols using 5-shot, 10-shot, and 20-shot settings, where n-shot refers to n labeled nodes per class for training. We use 500 nodes for validation and 1,000 nodes for testing across all datasets to ensure consistency with prior work [8, 47]. For the larger ArXiv dataset, we evaluate model performance under varying supervision levels using label ratios of 1%, 2.5%, and 5%. We adopt the original split for both validation and testing [21]. Due to computational constraints during inference, we randomly sample 30,000 nodes from the remaining unlabeled set for pseudo-label generation and model training.

GNN Component. For the GNN component of our GNN-as-Judge framework and other GNN-based baselines, we implement a 2-layer GCN architecture [27] with 64-dimensional hidden representations. We perform a limited grid search on dropout rates, exploring values in [0.3, 0.5, 0.7], and include batch normalization in our architecture. We set the learning rate to 1e-2 and train for up to 500 epochs with a patience of 100 for early stopping. For optimization, we use the Adam optimizer with a weight decay of 5e-4, following standard practices in graph neural network training.

LLM Component. Our GNN-as-Judge framework utilizes two language models as base models: Llama-3-8B-Instruct [16] and Mistral-7B-Instruct-v0.2 [25]. For both models, we implement parameter-efficient fine-tuning using LoRA [20] with rank 8 and alpha value

Table 5: Label spaces of the datasets used in our experiments.

Dataset	Label Space
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

of 16, which enables efficient adaptation of the pre-trained weights while maintaining computational feasibility. We set the dropout rate to 0.1 and use a batch size of 8.

- *Instruction Tuning Configuration.* We apply model-specific learning rates and training schedules based on our empirical findings: We use a learning rate of 5e-6 for Llama-3 models. Training duration varies by dataset size: 15 epochs for small-scale datasets (Cora, Citeseer) and 3 epochs for the larger ArXiv dataset. For Mistral-7B, we use a slightly higher learning rate of 1e-5. The training schedule includes 15 epochs for Cora, Citeseer, and PubMed, while the larger ArXiv dataset requires only 3 epochs.
- *Weakly-Supervised Fine-Tuning Configuration.* Our GNN-as-Judge approach uses carefully tuned hyperparameters for each model: We maintain a learning rate of 5e-6 with 5 epochs for smaller datasets Cora, Citeseer, 3 epochs for medium size dataset Pubmed and 2 epochs for ArXiv. For weakly supervised fine-tuning with Mistral-7B, we use a learning rate of 1e-5 with 8 epochs for Cora and Citeseer, and 3 epochs for both PubMed and ArXiv. The hyperparameter λ in Eq. 5, which balances instruction tuning and preference tuning losses, is set to 0.1 across all datasets and settings based on our parameter sensitivity analysis.

Selection Hyperparameters. For our proposed *GNN-as-Judge* pseudo-label selection mechanism described in Section 3 and outlined in Algorithm 1, we employ several key hyperparameters. We use dataset-specific confidence thresholds (τ_{conf}): 0.3 for Cora and Citeseer, 0.95 for PubMed, and 0.7 for ArXiv, which were determined empirically based on GNN prediction confidence distributions. The class-aware selection ratio (β) is set to 0.8 across all datasets, retaining the top 80% most confident predictions within each class to maintain class diversity. For the disagreement selection proportion (γ), we use 0.4 for Cora and Citeseer, and 0.2 for PubMed and ArXiv, selecting different proportions of disagreement nodes based on dataset characteristics. To balance preference strength and structural importance in the combined score calculation, we set $\beta_1 = 0.5$ and $\beta_2 = 0.5$, giving equal weight to both factors. For computing structural importance, we employ the standard PageRank algorithm with a damping factor of $\alpha = 0.85$.

Computing Environment and Resources. The computational resources includes four NVIDIA A100 GPUs, each with 80GB of memory.

D Prompts

Illustration of Prompts

Cora Dataset:

Instruction: 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?

Answer: ⟨**Predicted Label**⟩

Citeseer Dataset:

Instruction: Given a node-centered graph with centric node description: ⟨**raw_text**⟩, each node represents a paper, we need to classify the center node into 6 classes: ⟨**labels**⟩, please tell me which class the center node belongs to?

Answer: ⟨**Predicted Label**⟩

PubMed Dataset:

Instruction: Given a node-centered graph with centric node description: ⟨**raw_text**⟩, each node represents a scientific publication, we need to classify the center node into 3 classes: ⟨**labels**⟩, please tell me which class the center node belongs to?

Answer: ⟨**Predicted Label**⟩

ArXiv Dataset:

Instruction: Given a node-centered graph with centric node description: ⟨**raw_text**⟩, each node represents a scientific paper, we need to classify the center node into one of 40 arXiv CS subcategories: ⟨**labels**⟩, please tell me which category the center node belongs to?

Answer: ⟨**Predicted Label**⟩

E Algorithm Details for GNN-as-Judge

F Case Study

To better understand why our *GNN-as-Judge* approach is effective, we conduct a case study on the Cora dataset analyzing how graph structural importance affects the predictive performance of both GNNs and LLMs. Figure 5 presents our analysis of model behavior specifically on disagreement cases—nodes where GNN and LLM

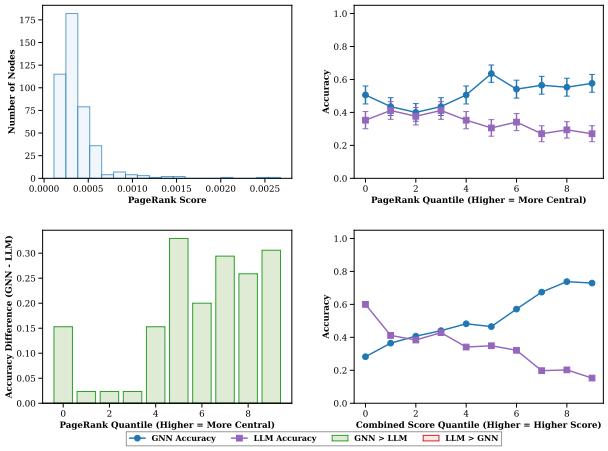
Algorithm 1 GNN-as-Judge for LLM Semi-supervised Learning on Graphs

Require: Labeled node set V_L, Unlabeled node set V_U, GNN model $\mathcal{M}_{\text{GNN}}$, LLM model $\mathcal{M}_{\text{LLM}}$
Require: Hyperparameters: confidence threshold τ_{conf}, selection ratios β, γ, preference weight λ, importance weights β_1, β_2
// Step 1: Train base models on labeled data
Training GNN $\mathcal{M}_{\text{GNN}}$ on labeled set V_L
Instruction tuning LLM $\mathcal{M}_{\text{LLM}}$ on labeled set V_L
// Step 2: Generate pseudo-labels for unlabeled nodes
for each node $v_i \in V_U$ **do**
  $\hat{y}_i^{\text{GNN}} \leftarrow \arg\max_j \mathcal{M}_{\text{GNN}}(v_i)_j$ // GNN prediction
  $\hat{y}_i^{\text{LLM}} \leftarrow \texttt{Parse}(\mathcal{M}_{\text{LLM}}(x_i))$ // LLM prediction
  $C_{\text{GNN}}(v_i) \leftarrow \max_j P_{\text{GNN}}(j|v_i)$ // GNN confidence
end for
// Step 3: Partition unlabeled nodes based on prediction agreement
$\mathcal{V}_{\text{agreed}} \leftarrow \{v_i \in V_U \mid \hat{y}_i^{\text{GNN}} = \hat{y}_i^{\text{LLM}}\}$
$\mathcal{V}_{\text{disagreed}} \leftarrow \{v_i \in V_U \mid \hat{y}_i^{\text{GNN}} \neq \hat{y}_i^{\text{LLM}}\}$
// Step 4: Select high-quality agreement nodes with class-aware filtering
$\mathcal{V}_{\text{filtered}} \leftarrow \{v_i \in \mathcal{V}_{\text{agreed}} \mid C_{\text{GNN}}(v_i) \geq \tau_{\text{conf}}\}$
$\mathcal{V}'_{\text{agreed}} \leftarrow \emptyset$
for each class $c \in \{1, \ldots, K\}$ **do**
  $R'_c \leftarrow \{v_i \in \mathcal{V}_{\text{filtered}} \mid \hat{y}_i^{\text{GNN}} = c\}$ // Nodes of class c
  $k'_c \leftarrow \lceil \beta \cdot |R'_c| \rceil$ // Number of nodes to select
  Add top-k'_c nodes from R'_c ranked by $C_{\text{GNN}}(v_i)$ to $\mathcal{V}'_{\text{agreed}}$
end for
// Step 5: Select informative disagreement nodes with structural importance
Compute PageRank scores $S_{\text{PR}}(v_i)$ for all $v_i \in \mathcal{V}_{\text{disagreed}}$
for each node $v_i \in \mathcal{V}_{\text{disagreed}}$ **do**
  $S_{\text{pref}}(v_i) \leftarrow P_{\text{GNN}}(\hat{y}_i^{\text{GNN}} \mid v_i) - P_{\text{GNN}}(\hat{y}_i^{\text{LLM}} \mid v_i)$
  $r_{\text{pref}}(v_i) \leftarrow \text{PercentileRank}(S_{\text{pref}}(v_i))$
  $r_{\text{PR}}(v_i) \leftarrow \text{PercentileRank}(S_{\text{PR}}(v_i))$
  $S_{\text{combined}}(v_i) \leftarrow \beta_1 \cdot r_{\text{pref}}(v_i) + \beta_2 \cdot r_{\text{PR}}(v_i)$
end for
$\mathcal{V}'_{\text{disagreed}} \leftarrow$ Top $\gamma \cdot |\mathcal{V}_{\text{disagreed}}|$ nodes ranked by S_{combined}
// Step 6: Weakly-supervised fine-tuning
Construct instruction tuning dataset: $\mathcal{D}_{\text{agreed}'} \leftarrow \{(x_i, \hat{y}_i^{\text{GNN}}) \mid v_i \in \mathcal{V}'_{\text{agreed}}\}$
Construct preference tuning dataset: $\mathcal{D}_{\text{disagreed}'} \leftarrow \{(x_i, \hat{y}_i^{\text{GNN}}, \hat{y}_i^{\text{LLM}}) \mid v_i \in \mathcal{V}'_{\text{disagreed}}\}$
Fine-tune LLM by minimizing:
$\mathcal{L}(\theta) = \mathbb{E}_{(x_i,y_i)\sim\mathcal{D}_{\text{agreed}'}}[\mathcal{L}_{\text{IT}}(\theta; x_i, y_i)] + \lambda \mathbb{E}_{(x_i,y_{w,i},y_{l,i})\sim\mathcal{D}_{\text{disagreed}'}}[\mathcal{L}_{\text{PT}}(\theta; x_i, y_{w,i}, y_{l,i})]$
return Fine-tuned LLM model $\mathcal{M}^*_{\text{LLM}}$

predictions differ—revealing when each model's predictions should be trusted.

The top-left panel shows the distribution of PageRank values across the Cora dataset, revealing a characteristic right-skewed distribution where most nodes have relatively low centrality, while a small number of nodes maintain higher PageRank scores. This aligns with the power-law distribution commonly observed in citation networks.

Figure 5: Analysis of structural importance and model performance on disagreement cases in the Cora dataset.



The top-right panel demonstrates how model accuracy varies across PageRank quantiles for disagreement cases. We observe a clear pattern: GNNs increasingly outperform LLMs as nodes become more structurally central in the graph. While both models achieve comparable accuracy for nodes with low centrality, their performance diverges significantly for highly central nodes. GNNs maintain robust accuracy cross higher centrality quantiles, whereas LLM performance deteriorates to 0.3-0.35, highlighting GNNs' superior ability to leverage structural information.

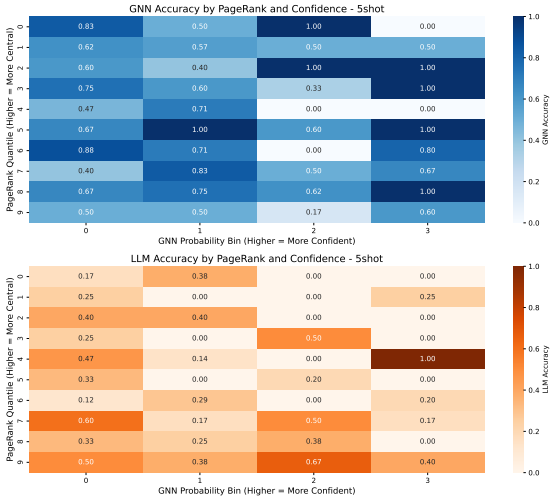
The bottom-left panel quantifies this performance gap by plotting the accuracy difference (GNN minus LLM) across PageRank quantiles. The consistently positive values demonstrate that for disagreement cases, GNNs provide more reliable predictions than LLMs across all structural importance levels. Notably, this advantage becomes more pronounced for highly central nodes, with accuracy differences of 0.2-0.3 in the higher quantiles. This confirms our hypothesis that GNNs derive particular advantage from graph topology when classifying structurally important nodes.

The bottom-right panel showcases the effectiveness of our combined score—a novel metric that integrates both structural importance and prediction preference. This visualization reveals a striking divergence in model performance:

- GNN accuracy shows a consistent, strong upward trend as the combined score increases, rising from approximately 0.3 to 0.75 across quantiles, confirming that our score effectively identifies nodes where GNNs excel.
- In contrast, LLM accuracy steadily deteriorates for higher-scored nodes, dropping from around 0.6 to approximately 0.15 for the highest quantiles, revealing the limitations of text-only approaches for structurally significant nodes.

These findings provide compelling empirical support for our *GNN-as-Judge* approach. The combined score effectively identifies cases where GNN predictions should be trusted over LLM outputs, particularly for nodes that are structurally important within

Figure 6: Comparison of GNN and LLM prediction accuracy across different PageRank quantiles and confidence levels in the 5-shot setting. Top: GNN accuracy tends to increase with both node centrality (higher PageRank) and prediction confidence. Bottom: LLM accuracy decreases significantly for nodes with higher PageRank values.



the graph topology. This analysis validates our pseudo-labeling strategy, which leverages the complementary strengths of both model types by using GNNs' structural awareness to guide LLM fine-tuning, especially for nodes where graph topology provides critical classification signals that pure text-based models cannot capture.

G Limitations

While *GNN-as-Judge* demonstrates strong performance, several limitations exist. First, *GNN-as-Judge* incorporates several hyperparameters related to pseudo-label selection and loss weighting that require selection to achieve optimal performance. For future work, developing automatic methods to select the optimal number of pseudo-labels and reducing hyperparameter dependence could address these limitations. Additionally, updating the model concurrently with pseudo-label generation may further improve performance.

H Broader Impacts

In this paper, we propose *GNN-as-Judge*, a framework that addresses challenges of utilizing LLMs for graph semi-supervised node classification. In real-world applications, labeled data are expensive and hard to obtain, thus our proposed approach could facilitate efficiency for many practical tasks such as social network analysis, citation network mining, and recommendation systems. By enabling more effective knowledge extraction from unlabeled nodes, our method could significantly reduce annotation costs and improve model performance when working with limited supervision.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009