LLMs Can Leverage Graph Structural Information in Text-Attributed Graphs

Anonymous authors
Paper under double-blind review

Abstract

A recurring claim in recent LLM-as-predictor work on text-attributed graphs (TAGs) is that in-context learning (ICL) benefits mainly from the textual attributes of neighboring nodes (often via homophily), while general-purpose LLMs cannot reliably exploit graph structure—especially edge direction and local topology. This paper re-evaluates that claim by asking a focused question: Can general-purpose LLMs genuinely leverage graph structural information in TAGs via ICL, once we remove confounding factors and provide an architecture explicitly designed for structural reasoning? We first introduce controlled neighborhood rewiring tests that keep node texts and label distributions fixed while perturbing structure. Across seven LLMs and four low-homophily WebKB graphs, both first-order flipping and two-hop extreme rewiring consistently degrade accuracy ($-2.06 \sim -23.15\%$ average relatively drop), demonstrating genuine structural sensitivity. After flipping, structural sensitivity strongly increases with model capability, and the performance advantage of stronger models arises primarily from correct structure rather than better text-only processing. We further show that apparent "structure misuse" in weaker models can be corrected by adding explicit step-by-step instructions. The previous claims is due to confounding factors—the traditional ICL framework lacks a dedicated mechanism for graph structure reasoning and handling lengthy multi-hop neighborhood contexts, rather than the inherent nature of LLMs themselves. Motivated by these findings, we propose the Text Attributes Passing Thoughts Network (TAPTN), an edge-aware, MPNN-like ICL framework that iteratively summarizes multi-hop neighborhoods using a structure-aware template and self-generated instructions. TAPTN substantially outperforms zero-shot CoT and GraphICL-style baselines on five TAG datasets by at least $+13.98\%$, especially on malignant heterophilic graphs (with $+15 \sim +25\%$ gain), and when used to produce structurally enriched texts for downstream fine-tuning, achieves performance competitive with state-of-the-art GNN pipelines. Collectively, the results establish that LLMs can exploit structure information in TAGs as effective as SOTA GNNs through ICL once with an appropriate architecture mitigating the confounding factors.

1 Introduction

Text-attributed graphs (TAGs), in which each node is associated with a rich textual description and edges encode relations between entities, are a fundamental abstraction for many real-world systems, including citation networks, e-commerce platforms, and financial transaction networks. A central task on TAGs is node classification, where the goal is to predict node labels by leveraging both node text and graph structure. Traditionally, LLM-based methods rely on fine-tuning (Zhao et al., 2025; Tang et al., 2024; Ye et al., 2024). Recently, large language models (LLMs) have emerged as powerful universal learners, and a growing body of work has explored whether LLMs can replace or complement graph neural networks (GNNs) on such graph reasoning tasks via in-context learning (ICL) rather than task-specific training due to its domain transferability and zero-shot learning capability. To integrate graph structural information, open-source LLMs like Llama can fuse Graph Neural Networks (GNNs) (Sun et al., 2025; Tian et al., 2023; Qin et al., 2023; Wang et al., 2024), while more powerful closed-source models like GPT-5.2 use In-Context Learning (ICL) for direct predictions (Chen et al., 2024; Guo et al., 2023) or additional insights (He et al., 2023).

Existing empirical evidence, however, has led to a prevailing misconception: LLMs are believed to benefit from neighboring nodes in TAGs primarily because of homophily in textual attributes with edges serving primarily as a retrieval mechanism for additional similar texts, rather than any genuine utilization of graph structure. Huang et al. (2024) systematically studied whether LLMs "can effectively leverage graph structural information through prompts," concluding that while LLMs do benefit from incorporating neighborhood information, the gains are strongly correlated with local homophily: nodes with neighbors sharing similar labels benefit most, and there is limited evidence that performance improvements stem from deeper structural reasoning beyond textual similarity. More recently, GraphICL (Sun et al., 2025) introduced a comprehensive ICL benchmark for graph reasoning with a unified prompt template that combines anchor-node text, task description, structure-aware neighbor texts, and labeled demonstrations. Their ablation studies show that the dominant source of improvement often comes from carefully selecting homogeneous neighbors and demonstrations (e.g., most similar or class-aligned nodes), whereas simply adding more structure-aware content yields smaller incremental gains.

Yet in many TAGs, edges and topological structures themselves are semantically meaningful. In financial transaction graphs, for example, specific topological patterns—such as multi-hop money-laundering chains, dense fraud rings, or temporal motifs—are strongly correlated with fraudulent behavior, even when the textual attributes of participating accounts are heterogeneous (Wei & Lee, 2025; Luo & Zhang, 2024; Tong & Shen, 2023). Similarly, in heterophilic graphs, neighbors often carry different labels, and what matters is not merely that a neighbor exists, but which role a neighbor plays in the local structure (buyer vs. seller, authority vs. hub, citing vs. cited). These observations suggest that graph structure information should be understood as more than "extra node texts": it includes edges and topological structures as relational evidence, providing context-dependent signals that determine how neighboring node information should be interpreted. If the misconceptions in existing researches are corrected, when and how LLMs utilize graph structure information in in-context learning (ICL) are understood, and the necessary architectures to exhibit this capability are identified, then the application areas of graph learning based on ICL will be greatly expanded.

We argue that two limitations in current ICL-based, LLM-as-Predictor approaches conspire to mask LLMs' latent ability to use such structural information, rather than indicating a lack of this capability in the LLMs themselves:

- **Homophily-centric interpretation of neighborhood gains**: Because prior analyses largely rely on high-homophily graphs and aggregate neighbors by simple textual concatenation, improvements from neighborhood information can be explained away as coming solely from label- and text-homogeneous neighbors, rather than from any genuine reasoning over graph structure. Under these settings, ablation experiments based on structural perturbations are unlikely to eliminate this misconception, because the effect of homophilous semantic redundancy (similar texts with similar labels) with the effect of graph connectivity can be conflated, making it difficult to determine whether LLMs are actually leveraging edges as relational signals.

- **Lack of a dedicated mechanism addressing lengthy high-order neighborhood contexts**: Existing ICL prompts typically linearize large ego-networks by listing first- and second-order neighbors verbatim. This leads to long, noisy contexts in which fine-grained structural cues (e.g., which neighbor is connected through which relation, or how multiple neighbors form a motif) are diluted. Long-context studies show that LLMs tend to attend most strongly to information at the beginning and end of the context, while information in the middle is disproportionately ignored, a phenomenon known as "lost in the middle" (Das et al., 2023; Liu et al., 2024). As a result, subtle structural patterns encoded in the interior of long neighborhood descriptions are unlikely to be consistently exploited, even if they are present.

- **Lack of a dedicated mechanism for structural reasoning:** Existing ICL-based TAG methods rely on prompt templates that treat neighbors as additional textual evidence or as labeled examples,

> but do not provide an explicit mechanism for representing and propagating edge information or multi-hop structure. Under this setting, as we found in Section 2, although LLMs can utilize graph structure information, some of this information is discarded or even misinterpreted, leading to the appearance that structural information is unhelpful or even detrimental to performance. Therefore, the correct way to utilize graph structure information should be to extract it from the LLM's parameter knowledge base and explicitly provide it to guide the LLM to consistently follow it.

In this work, we revisit the central question: **Can general-purpose LLMs genuinely leverage graph structural information in text-attributed graphs via ICL, once we remove these confounding factors and endow them with an architecture explicitly designed for structural reasoning?**

Our answer proceeds in three stages:

First, in Section 2, we re-examine the conclusion that "LLMs cannot effectively leverage graph structure via ICL" by revisiting the experimental setup that led to it. Rather than relying on high-homophily citation graphs with verbose second-order neighbor descriptions, we use directed, low-homophily datasets and design rewiring schemes that perturb first-order neighborhoods while keeping node texts fixed. This setting prevents good performance from being explained purely by treating neighbors as a bag of similar texts. Evaluating seven modern LLMs (including GPT-3.5, Phi, Gemma, Llama-3 family, and Qwen) on four heterophilic datasets, we find that all models exhibit systematic accuracy drops ($-2.06 \sim -23.15\%$ average relatively drop), providing direct evidence that LLMs are capable, in principle, of using graph structure information when prompts do not conflate structure with homophily. Experimental results also show that the sensitivity of LLMs to graph structure perturbations is positively correlated with their capabilities, and that better utilization of graph structures is the main source of their performance gains.

Second, in Section 3, motivated by the above diagnosis, we propose the Text Attributes Passing Thoughts Network (TAPTN), an ICL architecture explicitly designed to expose and exploit graph structure for node classification. TAPTN is inspired by Message Passing Neural Networks (MPNN) (Gilmer et al., 2017): it iteratively extracts and aggregates neighborhood information in textual form, guided by LLM-generated step-by-step instructions, but processes only first-order neighborhoods at each iteration. Instead of simply concatenating all neighbor texts, TAPTN constructs structured, layer-wise summaries of a node's multi-hop neighborhood, which simultaneously incorporates the text attributes of neighboring nodes, edge semantics, and the semantics of specific topological structures. This design addresses the two major shortcomings of existing ICL-based node classification methods: it decouples multi-hop reasoning into a sequence of shorter, structurally grounded steps (mitigating long-context issues) and provides a dedicated mechanism for reasoning over edges and connectivity. TAPTN thus provides, to the best of our knowledge, the first ICL-based LLM-as-Predictor architecture for TAG node classification that is explicitly engineered to leverage graph structure, not just treating neighbors as undifferentiated extra text.

Third, in Section 4, we revisit the comparison between LLM-based architectures and GNNs from a structural perspective. We fine-tune a small pre-trained language model on TAPTN-enhanced text attributes and compare its node classification performance to state-of-the-art GNN baselines on both homophilic and heterophilic TAGs. Rather than asking whether a naive zero-shot prompt can match a fully supervised GNN, we ask whether, once equipped with a reasonably designed ICL architecture that foregrounds structural information, LLMs' structural utilization capability is still inferior to that of GNNs.

Our contributions are summarized as follows:

First, we show that prior negative or homophily-centric conclusions are largely artifacts of high-homophily datasets and verbose, poorly structured prompts. Under low-homophily, structurally perturbed settings, LLMs exhibit clear sensitivity to graph structure, even when node texts attributes are held fixed, revealing LLMs' graph structure exploiting ability. Under flipping rewiring settings, structural sensitivity is strongly and significantly correlated with model capability, indicating that stronger models derive more of their gains from edge-based reasoning rather than from homophilic neighbor texts.

Second, we introduce TAPTN, a structure-aware ICL architecture for TAG node classification addressing lengthy high-order neighborhood contexts and enabling dedicated structural information leveraging mechanism. Considering both homophilic and heterophilic TAG benchmarks, TAPTN consistently outperforms zero-hop chain-of-thought and GraphICL-style baselines by at least +13.98%. Ablation studies show that these gains persist after controlling for self-reflection and are amplified by explicit step-by-step instructions. This result indicate that with appropriate framework like TAPTN, LLMs can indeed exploit graph structure information effectively.

Third, we show that a modest LM trained on TAPTN-generated textual representations achieves node classification performance competitive with state-of-the-art GNNs on both homophilic and heterophilic TAG benchmarks. This indicates that, when supported by an appropriate ICL architecture, LLMs' ability to exploit graph structure, including edges and local topology, matches or surpasses that of specialized graph neural networks rather than inherently weaker.

2 Are LLMs Really Unable to Leverage Graph Structural Information through ICL?

n this section, we revisit the claim that LLMs "cannot effectively leverage graph structural information through ICL and only treat neighbors as linear text," originally drawn from experiments on high-homophily citation graphs with long, verbose neighborhood descriptions. As discussed in the introduction, such settings make it difficult to distinguish whether performance gains come from (i) genuine use of graph structure (edges and their patterns), or (ii) graph homophily and textual similarity between neighbors. Moreover, long concatenated neighborhood descriptions are precisely the regime where LLMs tend to be "lost in the middle" and under-utilize information buried in the middle of the context.

Our goal here is to design an evaluation that removes these confounding factors and asks a sharper question:

When we keep node texts fixed and modify only edges in low-homophily graphs, do LLM predictions change in a systematic way?

A positive answer would provide direct evidence that LLMs interpret neighborhood information as graph structural data rather than as an unordered bag of texts, because any change in accuracy must arise from how the LLM interprets edge patterns.

2.1 Preliminary: homophily and the role of graph structure

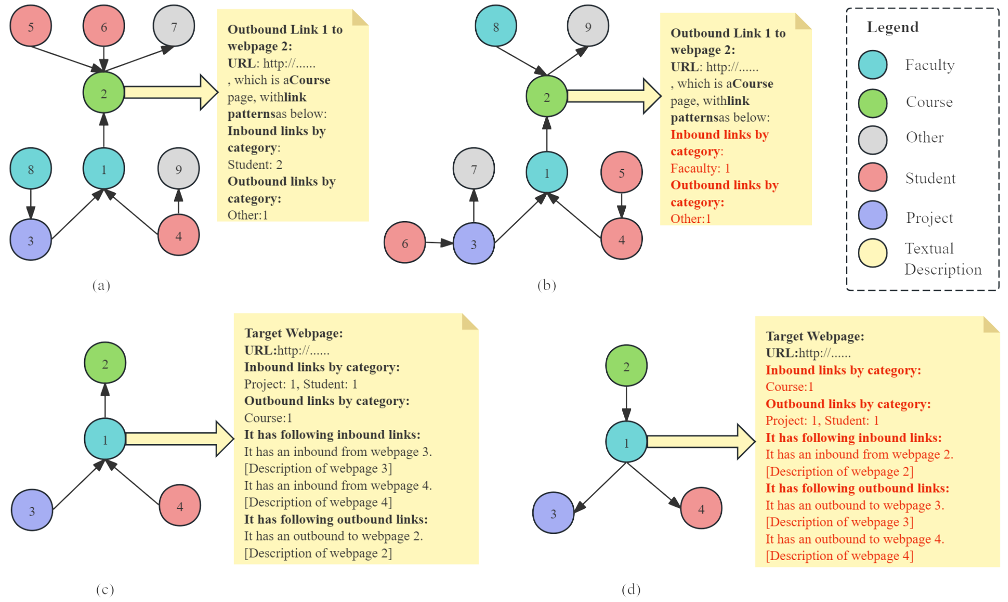
To measure homophily for a graph $\mathbf{G} = \{\mathbf{V}, \mathbf{E}\}$, we define it as:

$$H = \frac{\sum_{i \in \mathbf{V}} \frac{\left|\{j | e_{ij} \in \mathbf{E}, y_i = y_j\}\right|}{\left|\{e_{ij} | e_{ij} \in \mathbf{E}\ \}\right|}}{|\mathbf{V}|}, \tag{1}$$

where e_{ij} is the edge between node i, j, $\mathbf{E}, \mathbf{V}$ are the edges, nodes set of $\mathbf{G}$ respectively, y_i is the label of node i. This formula measures the average proportion of a node's neighbors sharing its label. Table 1 presents the homophily values for each dataset. The citation datasets used by Huang et al. (2024) exhibit high homophily ($H \geq 0.63$), whereas the WebKB datasets we use (Cornell, Texas, Washington, Wisconsin) have substantially lower homophily ($H \leq 0.19$).

On high-homophily graphs, an LLM can achieve good node classification accuracy by effectively treating neighbors as extra similar texts and ignoring graph structure. This is exactly the mechanism highlighted in previous ablation studies (Huang et al., 2024; Sun et al., 2025), where performance gains are largely attributed to carefully selecting homogeneous neighbors and demonstrations rather than to sensitivity to edge patterns or multi-hop topology.

However, in heterophilic graphs, as neighbors often have different labels and graph structure carry semantics and determine how neighbor information should be interpreted, what matters is which node links to which through which relationship, not merely that a neighbor exists. For example, in WebKB, hyperlink patterns between student, faculty, department, and course pages encode roles and authority relationships that are not recoverable from page texts alone.



Outbound Link 1 to webpage 2:
URL: http://......
, which is a**Course** page, with**link patterns**as below:
Inbound links by category:
Student: 2
Outbound links by category:
Other:1

Outbound Link 1 to webpage 2:
URL: http://......
, which is a**Course** page, with**link patterns**as below:
Inbound links by category:
Facaulty: 1
Outbound links by category:
Other:1

Target Webpage:
URL:http://......
Inbound links by category:
Project: 1, Student: 1
Outbound links by category:
Course:1
It has following inbound links:
It has an inbound from webpage 3.
[Description of webpage 3]
It has an inbound from webpage 4.
[Description of webpage 4]
It has following outbound links:
It has an outbound to webpage 2.
[Description of webpage 2]

Target Webpage:
URL:http://......
Inbound links by category:
Course:1
Outbound links by category:
Project: 1, Student: 1
It has following inbound links:
It has an inbound from webpage 2.
[Description of webpage 2]
It has following outbound links:
It has an outbound to webpage 3.
[Description of webpage 3]
It has an outbound to webpage 4.
[Description of webpage 4]

Figure 1: Illustration of "extreme" and "flipping" rewiring methods applied to a webpage neighborhood structure, with node 1 as the target node. Sub-figures (a) and (b) show "extreme" rewiring effects, while (c) and (d) depict "flipping" rewiring effects. Bold text highlights basic description components, and red text indicates changes post-rewiring. The target node's label is excluded from the second-order neighborhood description.

Table 1: Homophily of datasets used to evaluate the impact of neighborhood rewiring in this paper (upper half) and Huang et al. (2024) (lower half).

Dataset	Cornell	Texas	Washington	Wisconsin	OGBN-Arxiv	Pubmed	Cora
Homophily	0.1308	0.1448	0.1599	0.1869	0.6358	0.7924	0.8252

2.2 Experiments

To remove the confounding factors and reassess the core claim of previous researches that LLMs "simply treat neighborhood descriptions as linear text" and cannot effectively leverage graph structural information through ICL. We design experiments with the following features:

- Use low-homophily graphs, where simple homophily-based majority voting is not sufficient.
- Manipulate edge patterns while holding node texts fixed, so any accuracy changes must come from how the LLM interprets changed connectivity (i.e., structure and relations), not from changed textual attributes.
- Avoid extremely long neighborhood prompts that bury structural cues in the middle of the context, which is known to reduce their impact due to "lost in the middle" behavior of LLMs on long inputs.

2.2.1 Datasets and Settings

We select four low-homophily WebKB (Ghani et al., 2001) graphs—Cornell, Texas, Washington, Wisconsin—with homophily between 0.13 and 0.19. These datasets consist of webpages from university CS departments, with nodes labeled as student, faculty, staff, department, course, or project. We exclude the "other" category from evaluation to avoid diluting accuracy with pages whose hyperlink and content patterns are highly heterogeneous.

To reduce context length and avoid the “lost in the middle” effect on long prompts, we construct a GraphICL-style prompt template that include: (i) first-order neighbor (linked webpages) descriptions, including hyperlink direction, their URLs, content abstracts and labels; (ii) the number of second-order neighbors by category; (iii) link-count statistics (in/out degree). This template emphasizes graph structure (who links to whom by which relationship type in what pattern) rather than extra textual content.

To make it possible for the LLM to notice and process all relevant graph structural cues within a manageable context window while keeping node texts fixed across original and rewired graphs, we adopt and adapt the “extreme” rewiring from Huang et al. (2024) and introduce a new “flipping” rewiring that directly perturbs first-order link directions. Figure 1 illustrates these rewiring schemes and how they modify the neighborhood description in the prompt. Specifically:

- **Flipping**: We introduce a new method that reverses link directions of all first-order edges touching the target node. For example, if a student page originally links to a course page (outgoing), after flipping the course page has an incoming edge from the student page. This changes the incoming/outgoing pattern and role interpretation of each neighbor (who links to whom), while keeping the set of neighbors, their text descriptions and labels unchanged.
- **Extreme**: For each target node, we keep its original first-order neighbors fixed, but reconnect all its second-order neighbors to a single randomly chosen first-order neighbor. This rewiring dramatically changes the local 2-hop topology (e.g., funnels all paths through one neighbor) while preserving which nodes are in the 2-hop neighborhood and their labels. We represent second-order information only by category counts, not by verbose textual descriptions, to emphasize the change in path structure rather than added text.

These settings directly targets graph structure as relational evidence: if the LLM were using neighbors only as an unordered bag of texts (homophily-driven), both extreme and flipping should have little impact, especially given that category counts and node texts remain fixed. Substantial performance changes would therefore indicate that the model is sensitive to how neighbors are connected and what role they play in the hyperlink structure.

To assess whether this structural sensitivity is model-specific or holds more generally, we conduct our experiments on a panel of seven widely-used LLMs spanning a wide range of capabilities (with their text Arena scores in LMArena leader board from 1224-1319), parameter scale (from 9B-72B), and providers: GPT-3.5-Turbo-0125, Phi-4, Gemma-2-9B, Llama-3-70B-Instruct, Llama-3.1-70B-Instruct, Llama-3.3-70B-instruct, Qwen2.5-VL-72B-Instruct.

2.2.2 Results

Table 2 shows the effects of the two rewiring methods on ICL-based node classification accuracy across 4 datasets and 7 models.

For flipping rewiring, all models show consistent performance degradation across the all WebKB datasets. Averaged over all models and datasets, the relative performance change is -10.67% ($p < 0.01$ with t-test). The strongest sensitivity is observed for Qwen-2.5-VL-72B, whose average accuracy decreases from 71.28% to 54.62%, a -23.15% relative drop, while the weakest sensitivity is observed for GPT-3.5-turbo-0125, where accuracy changes from 46.67% to 45.89% on average, i.e., -2.06% relatively. These drops cannot be attributed to losing homophilic neighbors or to changing the textual attributes of the context. Instead, they show that all tested models rely on directional structure (whether a node mostly points to certain page types or is referenced by them) as a useful signal for classification. When we invert this signal, performance suffers. The sensitivity to flipping rewiring significantly strengthens for more capable LLMs.

For extreme rewiring, we again observe universal degradation, but with a different profile. The average relative performance change over models and datasets is -4.40%. Although it’s noticeably smaller in magnitude than under flipping, average accuracies of all models still drop across all datasets. The strongest sensitivity is seen for Gemma-2-9B (with -11.66% relative drop) while the weakest sensitivity appears for Llama-3.3-70B-instruct (-3.09% relative drop) . Thus, perturbing 2-hop topology while keeping 1-hop neighbors fixed still

Table 2: Impact of neighborhood rewiring on node classification accuracy (%), where $\mathbf{O_F}, \mathbf{R_F}, \mathbf{O_E}, \mathbf{R_E}$ are accuracy before and after flipping and extreme rewiring respectively. With 54 of total 56 settings, rewiring significantly decreases accuracy, indicating LLMs incorporate graph structural information despite their intelligences, parameter scales or providers.

Model	**Arena Score**	**#Parameters**	**Dataset**	$\mathbf{O_F}$	$\mathbf{R_F}$	$\mathbf{O_E}$	$\mathbf{R_E}$	$\mathbf{\Delta_F}$	$\mathbf{\Delta_E}$
GPT-3.5-Turbo-0125	1224	∼20B	Cornell	49.39	55.06	80.57	78.95	+6.67	−1.62
			Texas	43.87	39.53	69.57	64.82	−4.34	−4.75
			Washington	51.36	49.81	69.65	67.70	−1.55	−1.95
			Wisconsin	42.04	39.17	67.52	63.38	−2.87	−4.14
Phi-4	1255	14B	Cornell	51.52	43.15	63.31	58.07	−8.37	−5.24
			Texas	41.01	39.45	60.94	55.08	−1.56	−5.86
			Washington	53.01	52.63	69.17	60.53	−0.38	−8.68
			Wisconsin	49.84	48.60	65.73	60.75	−1.24	−4.98
Gemma-2-9B	1265	9B	Cornell	49.39	43.95	71.77	63.71	−5.44	−8.06
			Texas	44.53	35.94	67.58	55.86	−8.59	−11.72
			Washington	54.51	50.38	72.56	63.91	−4.13	−8.65
			Wisconsin	60.44	51.41	76.01	71.34	−9.03	−4.67
Llama-3-70B-Instruct	1276	70B	Cornell	73.79	52.02	82.66	83.06	−21.77	+0.40
			Texas	64.06	50.00	83.21	78.52	−14.06	−4.69
			Washington	67.67	58.27	84.96	77.82	−9.40	−7.14
			Wisconsin	71.65	59.81	84.42	83.80	−11.84	−0.62
Llama-3.1-70B-Instruct	1293	70B	Cornell	77.02	55.24	83.87	83.06	−21.78	−0.81
			Texas	69.53	52.73	84.38	80.47	−16.80	−3.91
			Washington	70.68	60.53	81.96	77.82	−10.15	−4.14
			Wisconsin	74.45	57.94	85.05	83.80	−16.51	−1.25
Qwen2.5-VL-72B-Instruct	1302	72B	Cornell	73.39	55.65	81.04	72.98	−17.74	−8.06
			Texas	69.92	48.44	76.95	73.04	−21.48	−3.91
			Washington	63.91	54.89	68.80	62.78	−9.02	−6.02
			Wisconsin	77.88	59.50	83.49	80.69	−18.38	−2.80
Llama-3.3-70B-Instruct	1319	70B	Cornell	75.40	51.61	78.63	77.02	−23.79	−1.61
			Texas	70.70	52.73	85.54	83.98	−17.97	−1.56
			Washington	69.55	62.03	77.07	74.06	−7.52	−3.01
			Wisconsin	74.45	60.44	81.93	78.19	−14.01	−3.74

harms performance for every model, but the effect is more uniform and generally weaker than for first-order flipping. Crucially, in both cases, node texts and label distributions are held constant. Performance differences between original and rewired graphs must therefore be attributed to graph structural information, not to different collections of homophilic neighbor texts.

Exceptionally, GPT-3.5-Turbo-0125 and Llama-3-70B-Instruct show modest improvements (+6.67% and +0.40%) with flipping and extreme rewirings on Cornell dataset respectively. These anomalous improvements suggests that LLMs sometimes misapplies structural knowledge: they appear to use link patterns, but not always in the correct way.

2.2.3 Is Structure Misapplication Caused by Knowledge Shortage or Inconsistent Adherence?

LLMs' occasional misapplication of graph structural information may result from prior knowledge errors or inconsistent correct method adherence due to lack of explicit instructions. To evaluate this, we appended

Table 3: Impact of flipping rewiring with step-by-step instructions on node classification accuracy of GPT-3.5-Turbo-0125. Rewiring consistently decreased accuracy across all datasets.

Dataset	**Original**	**Rewired**	**Δ(Acc.)**
Cornell	52.14	50.97	-1.17
Washington	61.54	59.92	-1.62
Wisconsin	50.96	49.04	-1.92
Texas	54.55	50.20	-4.35

LLM-generated step-by-step instructions to the prompts and re-evaluated performance after flipping rewiring with GPT-3.5-Turbo-0125.

As shown in Table 3, with explicit step-by-step instructions, structural rewiring decreased performance by over -1.17% on all datasets, averaging a -2.27% decrease. This demonstrates that LLMs, even weak LLMs such as GPT-3.5-Turbo-0125, possess correct methods for using graph structural information (it knows that who links to whom and how many neighbors of each type matter), but without explicit guidance, it does not consistently adhere to this method in its chain-of-thought, sometimes applying it in the wrong direction.

2.2.4 Structural Sensitivity and Model Capability

We now relate structural sensitivity under rewiring to model capability, as measured by LMArena scores. For a model M, we define its sensitivity S_R^M to a rewiring operation r as:

$$S_r^M = O_r^M - R_r^M, \tag{2}$$

where O_r^M, R_r^M are average accuracy over 4 datasets before and after rewiring r of M. Larger sensitivity indicates larger performance loss when structure is perturbed.

We examine how O_r^M, R_r^M and S_r^M vary with the LMArena score C^M of each model M through linear regression, as shown in Table 4, Figure. For flipping rewiring, we observe taht the model's sensitivity to rewiring is strongly and linearly correlated with its LMArena score (Pearson $r = 0.916, p = 0.0038$; Spearman $\rho = 0.893, p = 0.0068$), with a 100-point increase in the score corresponding to a $+19.43\%$ increase in sensitivity. The LMArena score also strongly explains the variance in sensitivity: a linear regression of sensitivity S_F on the LMArena score yields $R^2 = 0.8391$, meaning that within our group's score range (1224–1319), this regression can explain over 80% of the variance in S_F. This finding suggests that the ability of LLMs to utilize graph structures is highly correlated with intelligence, and more intelligent models possess a stronger ability in this regard. O_F and R_F also show credible linear positive correlations with the LMArena score (Pearson $p < 0.01$, $R^2 > 0.7$), with slopes of 3.369e-3 and 1.426e-3 respectively, demonstrating that more powerful models have a significantly greater advantage on graph tasks with reliable structural information than when such information is unavailable. However, for extreme rewiring, although all LLMs remain sensitive, the sensitivity lacks a statistically significant correlation with model capabilities.

Flipping directly changes the direction of first-order edges, which significantly impacts how node roles are interpreted (e.g., "student page points to multiple course pages" vs. "course page is referenced by multiple student pages"). This type of rewiring is highly informative for heterophilic hyperlink graphs, and powerful LLMs can easily read it when presented in the form of concise neighborhood descriptions. As model capabilities improve, LLMs can gain a deeper understanding and utilization of the graph structure, and the accuracy of this process highly depends on these directional cues. Therefore, when the graph structure is flipping rewired, more powerful models suffer greater performance losses. In contrast, extreme rewiring preserves all first-order edges and only perturbs two-hop connections, which we expose to the model through coarse category counts (e.g., counts of two-hop student/teacher/course neighbors). This representation is too coarse, and its compression method results in information loss. Even high-capacity models may only be able to use these descriptions in a relatively simple way, unable to translate their additional intelligence into more complex two-hop reasoning. Therefore, the sensitivity of models with different capabilities to this

Table 4: Correlation and regression between model capability (LMArena score) and performance under rewiring on WebKB. Regression slopes are reported per 100 LMArena points.

Target	Statistic	Flipping	Extreme
S_r^M	Pearson r	$+0.9160$	-0.2521
	Pearson p	0.0038	0.5855
	Spearman ρ	$+0.8928$	-0.5714
	Spearman p	0.0068	0.1802
	R^2	0.8391	0.0635
	Intercept	-2.3737	0.2685
	Slope	$+0.1943$	-0.0176
	RMSE	0.0251	0.0199
Orig Acc	Pearson r	$+0.8999$	$+0.6422$
	Pearson p	0.0058	0.1199
	Spearman ρ	$+0.8929$	$+0.6429$
	Spearman p	0.0068	0.1194
	R^2	0.8099	0.4124
	Intercept	-3.6786	-1.0747
	Slope	$+0.3369$	$+0.1440$
	RMSE	0.0481	0.0507
Rewired Acc	Pearson r	$+0.8414$	$+0.5858$
	Pearson p	0.0176	0.1669
	Spearman ρ	$+0.7857$	$+0.6071$
	Spearman p	0.0362	0.1482
	R^2	0.7080	0.3432
	Intercept	-1.3049	-1.3433
	Slope	$+0.1426$	$+0.1616$
	RMSE	0.0270	0.0659

type of rewiring is relatively uniform. Additionally, when utilizing two-hop neighborhood information, the average accuracy of all models across 4 datasets is still higher than when only using one-hop neighborhoods, even when the second-hop graph structure is extreme rewired. The additional performance gain may come from utilizing the overall category distribution patterns of all second-order neighbors, which does not require high intelligence.

In summary, the results of the statistical analysis support the following two conclusions:

- **Structural sensitivity is an intelligence-dependent behavior.** Stronger models benefit more from having the correct edge directions and lose more when those directions are corrupted. That is, their performance improvement is not just "better text understanding" but better structural utilization.
- **Most of the extra gain of strong models comes from exploiting structure.** Because O_F rises significantly faster than R_F with model capability, the additional headroom of frontier LLMs manifests primarily on the true graph, not on the structurally corrupted one. This indicates that, on low-homophily WebKB graphs, a substantial portion of their improvement over weaker models comes from using edges and edge directions as predictive signals, rather than from reading neighbor texts alone.

2.2.5 Case Study

As shown in Figure 4, flipping rewiring alters the LLM's prediction of a student page to a staff page from Cornell dataset by reversing the causal logic of the graph structure, even when textual attributes remain identical.

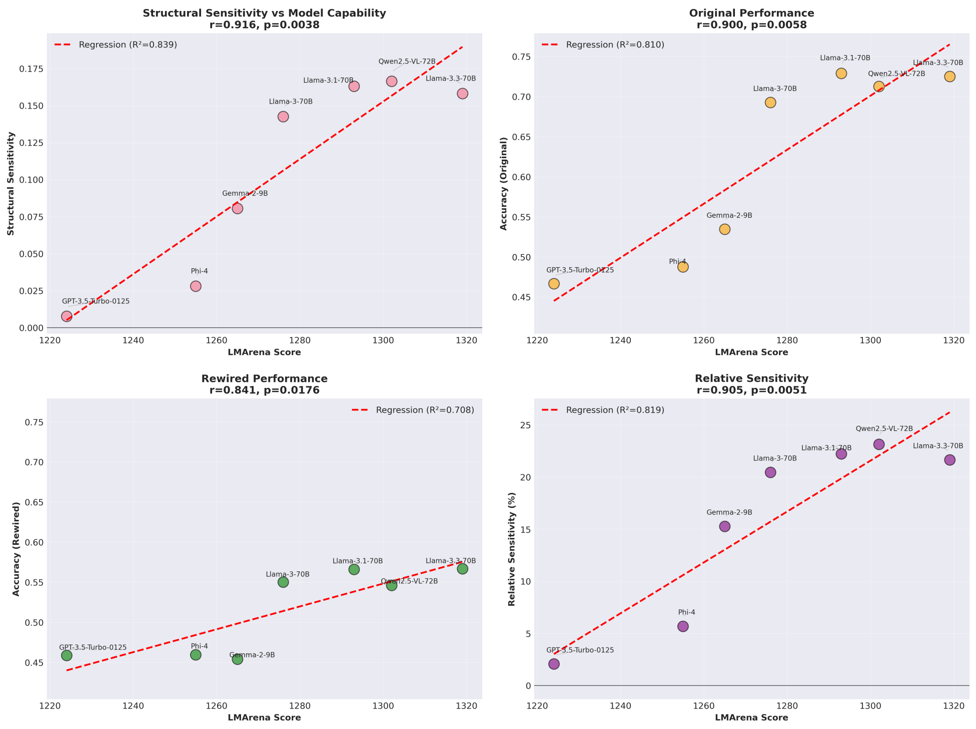


Figure 2: Correlation analysis between model capability and its sensitivity to flipping rewiring. (a) Stronger models show greater structural sensitivity (r=-0.92, p<0.01). (b-c) Unified y-axis scales reveal stronger models gain more from original structure than from rewired structure. (d) Relative sensitivity by percentage. All panels include regression lines (red dashed).

In the original graph, the target node receives an inbound link from a "Student Directory" page (".../students.html") and has outbound links to various "Staff" and "Project" pages. The model correctly interprets the inbound link from a directory as a membership signal ("The webpage has an incoming link from... a directory of students... This suggests that the webpage is related to a student"). In this reasoning process, LLMs correctly extracted the most informative clues for downstream tasks from the local topological structure, accurately understood and utilized the actual semantics implied by these clues, thereby achieving correct predictions, rather than relying on homophily.

After flipping rewiring, all edge directions are reversed. The target node now points outward to the "Student Directory" and receives inbound links from "Staff" and "Project" pages. The model re-interprets the target's role. It views the target as a hub or authority that is referenced by staff members and provides resources (links) to the student directory ("Outgoing link to a directory... suggests a connection to the academic community... possibly a staff member's profile"). The model still attempts to extract clues from the topological structure and understand its semantics, but misinterpreting the structural role due to the reversed edges thus changes its prediction to Staff (with 0.8 confidence).

This failure mode confirms that LLMs do not merely treat neighbors as a "bag of words" or rely on homophilic heuristics that neighboring nodes sharing the same label. If they did, the prediction would remain unchanged because the neighbor texts were identical in both scenarios. Instead, the specific direction of the edges (Inbound vs. Outbound) acted as a critical semantic signal for defining the node's role.

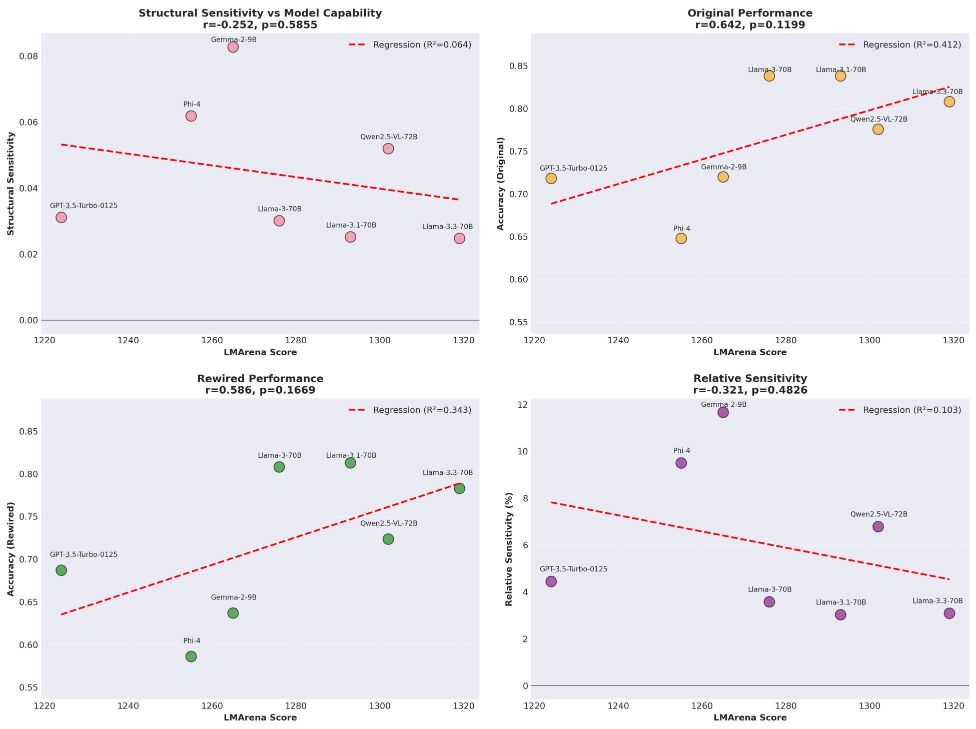


Figure 3: Correlation analysis between model capability and its sensitivity to extreme rewiring. (a) No correlation between capability and structural sensitivity (r=-0.25, n.s.), indicating uniform dependence. (b-c) Unified y-axis scales show nearly identical slopes for original and rewired performance, contrasting with Hop=1. (d) Relative sensitivity shows no capability-based pattern.

3 How to Leverage Graph Structural Information More Effectively through ICL?

3.1 Text Attribute Passing Thoughts Network

In this section, we aim to propose a novel ICL framework that eliminates the key shortcomings that led previous studies to conclude that LLMs cannot utilize graph structures, and demonstrate that this framework achieves additional performance gains from graph structures compared to traditional ICL methods that only utilize homogeneity and neighbor attributes, Furthermore, and it's also effective for heterophilic graphs. An affirmative answer will further support the core argument of this paper: LLMs genuinely leverage graph structural information in text-attributed graphs via ICL, once we remove these confounding factors and endow them with an architecture explicitly designed for structural reasoning.

In node classification tasks, ICL faces two key limitations:

Lack of an explicit structural reasoning mechanism:

- **Template without structure:** Prior ICL-based TAG methods tend to treat neighbors either as extra textual evidence or as labeled exemplars, without an explicit representation of edge semantics or multi-hop topology.
- **Inconsistent reasoning:** Flipping-rewiring experiments showed that LLMs already possess a correct method for using structural information (e.g., interpreting hyperlink direction and local link patterns). However, without explicit guidance, LLMs, especially low-capability model, sometimes fail to apply this method consistently (e.g. GPT-3.5-Turbo-0125 uses structural cues in the wrong direction on the Cornell dataset anomaly before adding instructions), leading to unstable performance.

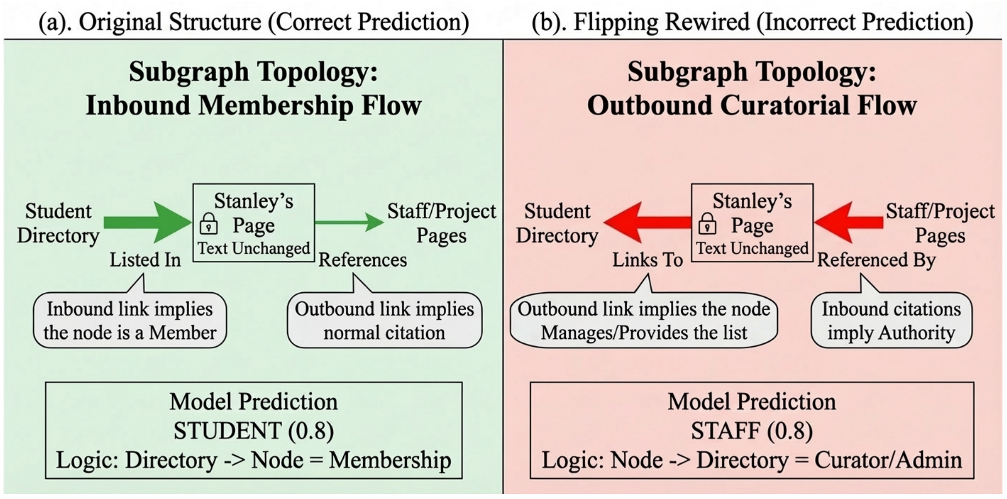

(a). Original Structure (Correct Prediction)
(b). Flipping Rewired (Incorrect Prediction)
Subgraph Topology: Inbound Membership Flow
Student Directory
Listed In
Stanley's Page
Text Unchanged
References
Staff/Project Pages
Inbound link implies the node is a Member
Outbound link implies normal citation
Model Prediction
STUDENT (0.8)
Logic: Directory -> Node = Membership
Subgraph Topology: Outbound Curatorial Flow
Student Directory
Links To
Stanley's Page
Text Unchanged
Referenced By
Staff/Project Pages
Outbound link implies the node Manages/Provides the list
Inbound citations imply Authority
Model Prediction
STAFF (0.8)
Logic: Node -> Directory = Curator/Admin

Figure 4: Case Study of flipping rewiring from Cornell dataset. This comparison illustrates how reversing edge directions alters the LLMs' interpretation of social roles, despite identical node text. (a) In the original graph, an inbound link from the "Student Directory" correctly signals membership (Prediction: Student). (b) After flipping rewiring, the edge is reversed; the node now links out to the directory, which the LLM re-interprets as a curatorial or administrative role (Prediction: Staff). This demonstrates that the model relies on structural directionality, not just textual homophily, for classification.

Lack of a dedicated mechanism addressing lengthy high-order neighborhood contexts: Existing ICL-based methods typically linearize large K-hop neighborhoods as long, noisy lists of raw node descriptions. In such settings, subtle structural cues (who links to whom, directionality, etc.), even it exist in template, are easily drowned out because of "lost in the middle" effect (information near the center of long prompts is used much less effectively than information at the beginning or the end). Conversely, as revealed by extreme rewiring experiments, if we attempt to compress high-order neighborhood information into a length that can be efficiently utilized, but with significant information loss, the benefits the model can gain are limited. This dilemma makes it difficult for the model to reliably utilize high-order structural signals unless we can summarize high-order neighborhood information, including neighbor node attributes and graph structure, with low information loss.

To integrate the structural information into template, when construct a structured textual description of a node's 1-hop neighborhood, we designing a Structure-Aware Template (SAT) containing the node's own attributes, neighbor attributes, edge directions and semantics (and node/edge types as an optional component), with its details in Appendix B. Compared to GraphICL-style template, our SAT emphasizes edges and topological structure as relational evidence: we explicitly describe which nodes link to which others and in what role, rather than simply listing neighbor texts as an unordered bag. To address the verbose, noisy context, we design a MPNN-like approach to extract effective information from K-order neighbors and aggregate it into (K-1)-order neighbors iteratively, generating enhanced text attributes. Over K iterations, LLMs can utilize K-order neighborhood information by processing only first-order neighborhood text descriptions in each iteration. This mechanism also allows task-relevant multi-hop topological patterns (such as motifs, directed flows) to be detected, extracted, and emphasized in the descriptions during the aggregation process. To avoid inconsistency, we introduce self-generated step-by-step instructions to guide LLM consistent adhere to correct graph structures usage in each iteration.

We propose the Text Attributes Passing Thoughts Network (TAPTN), a new ICL method for node classification, as illustrated in Figure 5. Like MPNNs, TAPTN consists of message-passing and readout phases. During the message-passing phase, K iterations extract and summarize information from the first-order neighborhood to enhance the node's text attributes. This increase enhanced text attributes' abstraction level, making them more relevant to the downstream task, and encapsulates K-order neighborhood information pertinent to the task, including structural features (such as anomalous trading patterns in financial

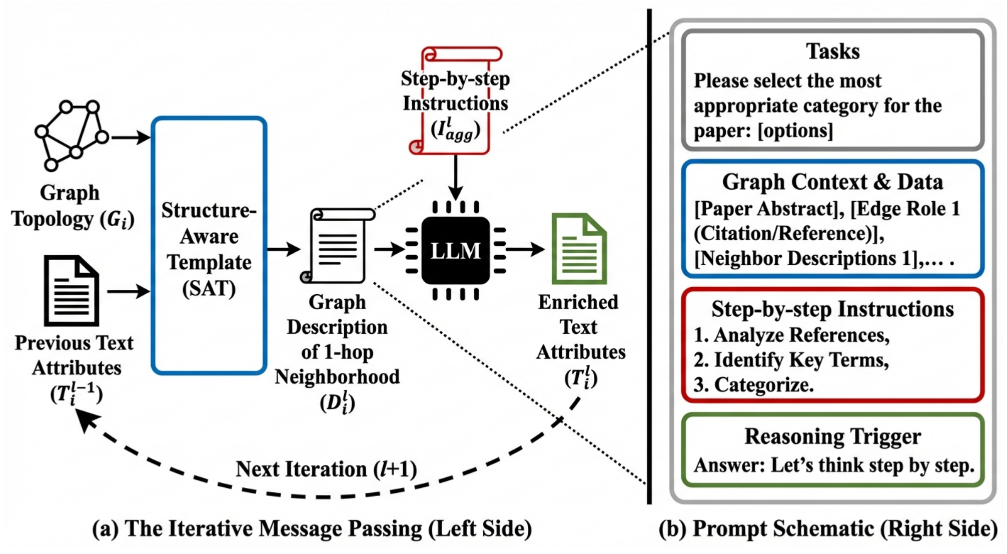

Graph Topology (G_i)
Previous Text Attributes (T_i^{l-1})
Structure-Aware Template (SAT)
Graph Description of 1-hop Neighborhood (D_i^l)
Step-by-step Instructions (I_{agg}^l)
LLM
Enriched Text Attributes (T_i^l)
Next Iteration (l+1)
Tasks
Please select the most appropriate category for the paper: [options]
Graph Context & Data
[Paper Abstract], [Edge Role 1 (Citation/Reference)], [Neighbor Descriptions 1],... .
Step-by-step Instructions
1. Analyze References,
2. Identify Key Terms,
3. Categorize.
Reasoning Trigger
Answer: Let's think step by step.
(a) The Iterative Message Passing (Left Side)
(b) Prompt Schematic (Right Side)

Figure 5: Subgraph (a) shows the iteration process of TAPTN. Subgraph (b) shows how the LLM-generated step-by-step instructions are attached to the prompt.

graphs), into an abstract. Unlike MPNNs, TAPTN passes text attributes instead of numerical tensors and relies on self-generated instructions based on LLMs' prior knowledge rather than minimizing a loss function on a training set, making it essentially a zero-shot learning method.

Specifically, the message-passing phase of the TAPTN framework is represented by:

$$T_i^l = LLM\left(T_i^{l-1}, D_i^{l-1}, I_{agg}^l, S_{agg}\right), \tag{3}$$

where T_i^l is the enhanced text attribute of node i after l iterations, starting with T_i^0, the node's original text attribute. I_{agg}^l is the LLM-generated instruction for the l-th aggregation that explicitly tells the model how to use neighbors and edges, and S_{agg} is the task description for the downstream task. D_i^l is the text description of node i's first-order neighborhood after l iterations, generated as:

$$D_i^l = F_{j\in N(i)}^{SAT}\left(G_i, m_{ij}^l\right), \tag{4}$$

where $N(i)$ is the neighborhood of node i, $F_{j\in N(i)}^{SAT}$ represents generating text description of $N(i)$ according to template SAT. Our designed SAT preserves the semantics of the edges, makes the utilization of edge information and the effectiveness on heterophilic graphs possible. For instance, in citation graphs it separates references and citations into distinct blocks; in e-commerce graphs it distinguishes co-purchased from also viewed edges; and in transaction graphs it can encode sender/receiver roles. G_i represents the topology structure of $N(i)$, and m_{ij}^l is the message sent from neighbor j to target node i during the l-th iteration of aggregation, generated as:

$$m_{ij}^l = F^{MSG}\left(T_j^l\right), \tag{5}$$

where F^{MSG} is the message generation function. In our experiments, similar to most MPNN variants, F^{MSG} is simply the identical function, but it can be adapted to different downstream tasks. For instance, if T_j^l is too lengthy, a summarization approach can be used, i.e.,

$$F^{MSG}\left(T_j^l\right) = LLM(T_j^l, I_{sum}^l, S_{sum}), \tag{6}$$

where I_{sum}^l is an instruction guiding LLMs to extract effective information, and S_{sum} is a task description for summarizing text attributes.

For the node classification task, the readout phase of TAPTN is defined by:

$$\hat{y}_i = F^{PAR}(T_i^K), \tag{7}$$

where F^{PAR} is a parser function extracting the predicted label $\hat{y}_i$ for node i from enhanced text attributes T_i^K after K iterations. While methods like regular expressions can implement F^{PAR}, due to their limited instruction-following capability, labels produced by LLMs may somtimes fall outside the candidate range or not be sorted by relevance even the reasoning process is correct. Therefore, we use the following F^{PAR}:

$$F^{PAR}(T_i^K) = LLM(T_i^K, S_{par}), \tag{8}$$

where S_{par} is the task description used to extract the node label. Step-by-step instructions, task descriptions, graph description generation templates and prompts templates in our experiments are detailed in Appendix B.

As a flexible framework, TAPTN can also be easily enhanced with an attention mechanism, implemented as follows:

$$W_{ij}^l = LLM(T_i^{l-1}, T_j^{l-1}, I_{att}^l, S_{att}), \tag{9}$$

$$D_i^l = F_{j \in N(i)}^{SAT}(G_i, m_{ij}^l, W_{ij}^l), \tag{10}$$

where W_{ij}^l indicates the importance of neighbor j to node i in the l-th iteration. I_{att}^l is the instruction guiding the LLM to calculate the importance of each neighbor, and S_{att} is the task description for this calculation. However, the attention mechanism is optional. As previous research has shown its effectiveness for ICL-based node classification, we did not use it in our experiments.

By incorporating these methods, TAPTN effectively addresses LLMs' challenges with overly verbose contexts and reasoning inconsistency through iterative extraction and aggregation guided by step-by-step instructions, improving performance in node classification tasks within a zero-shot ICL setting.

3.2 Experiments

To verify the effectiveness of TAPTN in addressing the shortcomings of existing ICL-based graph classification methods, and to demonstrate that graph structure can indeed be effectively understood and utilized by LLMs with these shortcomings mitigated by TAPTN framework, rather than just benefit from homophily and neighbor attributes, we tested two hypotheses:

H1: Stable performance gain: TAPTN consistently delivers performance gains exceeding the GraphICL-style baselines.

H2: Stable gain from structure on heterophilic graph: TAPTN consistently obtains performance improvements on heterophilic graphs after integrating neighborhood information, with larger neighborhood hops leading to greater gains.

If these two Hypotheses hold, TAPTN will necessarily be an effective framework with which LLMs can correctly understand and utilize graph structure information. Otherwise, TAPTN will fail to achieve stable performance superior to Naive Zero-Shot CoT which simply uses the target node's text attributes on heterophilic graphs lacking homophily.

We evaluate on five text-attributed graphs spanning two regimes from LLMNodeBed (Wu et al., 2025) benchmark:

- **Homophilic citation graphs:** Cora (Yang et al., 2016) and Arxiv-2023 (Huang et al., 2024). Due to the large number of nodes in Arxiv-2023, we selected subgraphs consisting of seven categories of nodes for the experiments.
- **Heterophilic webpage graphs:** Texas, Wisconsin, Cornell from WebKB (Ghani et al., 2001) dataset. These datasets fall in (or close to) the mid-homophily pitfall range $(0.1, 0.3)$ (Luan et al., 2024b) and are lassified as malignant heterophilic graphs considered particularly challenging for node classification (Luan et al., 2024a).

Table 5: Node classification accuracy (%) across five text-attributed graphs. GraphICL denotes the GraphICL-style baseline that concatenates neighborhood text without a dedicated mechanism for leveraging graph structure or addressing verbose high-order neighborhood context.

Method	**Cora**	**Arxiv-2023**	**Texas**	**Wisconsin**	**Cornell**
0-hop (zero-shot CoT)	59.40	76.19	66.93	68.85	72.58
GraphICL 1-hop	62.55	83.49	65.23	76.01	76.21
GraphICL 2-hop	63.47	85.40	66.41	70.72	72.18
TAPTN 1-hop	72.69	87.30	89.84	85.67	85.08
TAPTN 2-hop	73.80	88.57	91.80	87.23	86.69

For each dataset, we compare:

- **0-Hop (Zero-shot CoT).** A graph-agnostic ICL baseline that uses only the target node's own text with chain-of-thought prompting, without any neighbors.
- **1-Hop / 2-hop GraphICL.** The original GraphICL-style prompt that linearizes one-hop or two-hop neighborhoods into a single context, without TAPTN-style iterative aggregation. For a clear comparison between the structure information leveraging, the GraphICL baseline in this paper is an enhanced version with the same explicit structure-aware templates as TAPTN. To ensure a fair comparison, GraphICL employs a random neighbor sampling strategy, allowing it to access the same set of neighbors as TAPTN.
- **1-Hop / 2-hop TAPTN.** Our full method with structure-aware template, $K = 1$ or $K = 2$ message-passing iterations and LLM-generated step-by-step instructions for each aggregation step and for the final classification.

We select GPT-3.5-Turbo-0125 as the backbone model for citation graphs and Llama-3.3-70B-Instruct for WebKB graphs. More details of our experiment settings are provided in Appendix C and D.

Table 5 shows that TAPTN's advantage persists and becomes substantially larger on malignant heterophilic graphs. Across all five datasets, 2-hop TAPTN is the top-performing zero-shot ICL method. Averaged over datasets, 2-hop TAPTN reaches 85.62% accuracy, compared to 68.79% for 0-hop CoT, 72.70% for 1-hop GraphICL, 71.64% for 2-hop GraphICL and 84.12% for 1-hop TAPTN. These results correspond to 2-hop TAPTN improves by +16.83% on average over 0-hop CoT, by +12.92% on average over 1-hop GraphICL, by +13.98% over 2-hop GraphICL and by +1.50% over 1-hop TAPTN. On Texas/Wisconsin/Cornell datasets, 2-hop TAPTN improves over 2-hop GraphICL by +25.39%, +16.5%, and +14.51% respectively, and by +24.87%, +18.38%, and +14.11% over 0-hop CoT. Notably, on these datasets, GraphICL exhibits little benefit from expanding to 2-hop neighborhoods (and can even degrade relative to 1-hop), and on Texas dataset 1-hop and 2-hop GraphICL even degrade from 0-hop zero-shot CoT, while TAPTN gains consistently from neighborhood information, and its gain consistently rises when considered neighborhood expands to 2-hop.

The WebKB dataset exhibits low homogeneity and is near the mid-homophily pitfall, meaning that simple heuristics relying on "neighboring nodes sharing the same label" or "neighboring nodes having different labels" are not feasible. In this case, the gains of TAPTN compared to GraphICL cannot be simply explained by "reading similar neighbors"; they must reflect a better utilization of the graph structure (e.g., specific hyperlink patterns). This gain becomes even more significant with the introduction of multi-hop neighborhood information, which is sometimes detrimental to GraphICL, further demonstrating that TAPTN's performance advantage over GraphICL stems from its unique mechanism for utilizing structural information.

In summary, both hypotheses H1 and H2 are valid, and with TAPTN, our effective ICL framework mitigating shortcomings of traditional ICL for graph tasks, LLMs can indeed understand and utilize graph structure information. Previous research suggested that LLMs lacked this ability due to limitations of traditional ICL frameworks, rather than an inherent limitation of LLMs themselves.

3.3 Ablation Study

To demonstrate that TAPTN successfully mitigates the shortcomings of traditional ICL frameworks and achieves performance gains through its dedicated multi-hop neighborhood information processing mechanism and structural information reasoning mechanism, rather than just improves performance simply by providing LLMs with more opportunities for "self-reflection" through iterative neighborhood processing, we evaluate two hypotheses:

H3: Enhanced utilization of K-hop neighborhoods through MPNN-like processing: The reason of TAPTN improves classification accuracy by leveraging multi-hop neighborhood information more effectively than GraphICL-style baselines, which directly concatenate neighborhood text, is iterative information extraction and aggregation (MPNN-like) dedicatedly designed for verbose multi-hop neighborhood context addressing rather than mere self-reflection.

H4: Improved adherence through instructions: LLM-generated step-by-step instructions increase the consistency with which LLMs apply correct, structure-aware reasoning procedures, thereby contributing additional gains.

To prove H3, we compared the accuracy of the following approaches on Cora and Arxiv-2023 datasets with GPT-3.5-Turbo-0125:

- TAPTN incorporating first-order and second-order neighborhood information;
- GraphICL incorporating first-order and second-order neighborhood information after 1-2 iterations of self-reflection. That is, the model first produces a preliminary label and explanation, then is asked to reflect on and revise its answer.

For H4, we evaluated the performance gains from adding step-by-step instructions across 1-hop and 2-hop settings. This experiment is conducted with the same settings as Section 3.2.

As shown in Table 6, For Cora and Arxiv-2023, adding self-reflection to naive GraphICL sometimes helps slightly but also frequently harms. For example, on Cora, 1-hop GraphICL improves from 62.55% (no self-reflection) to 69.19% (with 1 iteration of self-reflection), but 2-hop GraphICL with self-reflection can underperform its non-reflective counterpart on Arxiv-2023, and the second iteration of self-reflection harms 1-hop GraphICL on both datasets. There is no consistent trend that "more reflection" yields monotonic gains. This comparison indicate that self-reflection alone is insufficient and inconsistent. TAPTN still dominates even after reflection. Even when naive GraphICL is given its best-performing RI configuration, TAPTN's 2-hop variant remains substantially better. Aggregating across datasets and configurations, TAPTN improves node classification accuracy by about +6.6% over the strongest self-reflective baseline. This confirms that TAPTN's advantages on multi-hop neighborhood information utilization stem from MPNN-like structured aggregation, not from merely allowing the LLM to re-think its answers more times.

Table 7 reports the impact of LLM-generated step-by-step instructions on TAPTN across five datasets and two neighborhood orders. Several consistent patterns emerge.

- **Instructions help on all datasets and all orders.** For every dataset and for both 1-hop and 2-hop TAPTN, adding instructions strictly improves accuracy; there is no case where instructions hurt performance. Averaged over all five datasets, instructions increase accuracy by +11.42% for 1-hop TAPTN and by +10.42% for 2-hop TAPTN.
- **On homophilic citation graphs, instructions are most beneficial at 1 hop.** On Cora and Arxiv-2023, instructions give sizable gains at 1 hop (+10.14% and +3.81%) but more modest gains at 2 hops (+2.95% and +1.27%). This aligns with our intuition that, when the first-order neighborhood already captures most of the useful homophilic signal, explicit procedural guidance mainly helps the model decide *which* immediate neighbors and edge roles to attend to, while additional guidance at 2 hops yields diminishing returns once a good 1-hop abstraction is formed.

Table 6: Comparison between TAPTN and GraphICL with self-reflection reported in percentage. "RI" denotes the number of self-revision iterations. TAPTN consistently outperforms naive GraphICL with self-reflection, with an average improvement of +6.59%, confirming the superiority of its framework in utilization of graph structural information.

Method	**Order**	**RI**	**Cora**	**Arxiv-2023**
GraphICL	1	0	62.55	83.49
		1	69.19	74.92
		2	68.45	73.65
	2	0	63.47	85.40
		1	69.18	75.87
		2	68.45	80.00
TAPTN	1	/	72.69	87.30
	2	/	**73.80**	**88.57**

Table 7: Impact of LLM-generated step-by-step instructions on node classification accuracy (%) for first and second-order neighborhoods on Cora and Arxiv-2023 datasets. Instructions significantly improved accuracy, especially for first-order neighborhoods.

Dataset	**Order**	**w/o instruction**	**w/ instruction**	**Δ(Acc.)**
Cora	1	62.55	72.69	+10.14
	2	70.85	73.80	+2.95
Arxiv-2023	1	83.49	87.30	+3.81
	2	87.30	88.57	+1.27
Texas	1	65.23	89.84	+24.61
	2	66.79	91.80	+25.01
Wisconsin	1	76.01	85.67	+9.66
	2	73.21	87.23	+14.02
Cornell	1	76.21	85.08	+8.87
	2	77.82	86.69	+8.87

- **On malignant heterophilic graphs, instructions are critical for both 1-hop and 2-hop reasoning.** For Texas, Wisconsin, and Cornell, instructions yield very large improvements, especially at 2 hops: Texas (+24.61 at 1 hop, +25.01 at 2 hops), Wisconsin (+9.66, +14.02), and Cornell (+8.87, +8.87). On these graphs, homophily is weak or misleading, so the model must rely on more subtle structural patterns spanning one and two hops. Without explicit instructions, TAPTN's iterative aggregation is not sufficient to consistently extract these patterns; with instructions, the model is guided to interpret edge roles and neighborhood composition in a task-specific way, turning latent structural sensitivity into substantial accuracy gains.

- **Instructions and the TAPTN architecture are complementary.** The fact that instructions are uniformly beneficial, and especially powerful on heterophilic datasets, reinforces our interpretation from Section 2: LLMs possess useful structural knowledge but often fail to apply it reliably. TAPTN's iterative, edge-aware message passing exposes the relevant structural cues, while the step-by-step instructions stabilize how the model uses these cues across iterations.

Table 8: Accuracy of TAPTN and GraphICL considering 1 or 2 order neighborhood on Amazon dataset. Even with Llama-3.1-instruct 7B for first iteration, 2-order TAPTN outperformed other methods, showing scalability of TAPTN's iterative aggregation and explicit guiding mechanisms.

Method	**Order**	**w/o instruction(%)**	**w/ instruction(%)**
Naive	1	74.75	/
	2	75.00	/
TAPTN	1	74.75	74.75
	2	74.00	**76.75**

TAPTN showed more significant improvements on the Cora dataset, where baseline accuracy was lower, indicating its particular effectiveness in tasks with ambiguous classification standards and overlapping node categories. Although there are concerns about reliance on instructions and potential error propagation, TAPTN's iterative process demonstrated resilience against cumulative errors. Since instructions are self-generated by LLMs, TAPTN remains scalable and practical without requiring manual intervention. A small annotated subgraph could further refine instructions if needed, enhancing accuracy and reducing risks. The specifically designed guidance mechanism is a core innovation overcoming LLMs' inconsistent adherence to correct graph structural methods.

In conclusion, the experiments confirm both hypotheses. TAPTN's iterative extraction and aggregation effectively utilize K-order neighborhood information, achieving higher accuracy than naive methods, even after self-reflection. Performance gains result from better utilization of high-order neighborhood structural information. Additionally, LLM-generated step-by-step instructions significantly enhance accuracy, particularly with first-order information, ensuring consistent adherence to correct graph structural usage. TAPTN overcomes LLMs' limitations with verbose contexts and inconsistent method application through ICL, broadening LLMs' applicability in node classification tasks by leveraging zero-shot learning and interpretability.

3.4 Scalability and Portability

As Cora and Arxiv-2023 are both graphs below 1000 nodes of citation network, concerns about TAPTN's scalability on large graphs and portability on different scenarios may arise. To demonstrate TAPTN's scalability and portability, we tested it on Amazon which is a e-commercial co-purchase network with 13482 nodes. Using large LLMs such as GPT-5 for large graphs may consume excessive computing resources, so we use Llama-3.1-instruct 8B for the first iteration of 2-order TAPTN while Llama-3.3-instruct 70B is used for the second iteration. As shown in table 8, 2-order TAPTN outperformed GraphICL, 1-order TAPTN (both with Llama-3.3-instruct 70B) and 2-order TAPTN without instructions by +1.75%, +2.00%, +2.75% respectively, showing TAPTN's iterative aggregation and explicit guiding mechanisms are scalable on large graphs and portable on different scenarios. Availability of small LLMs for the first iteration makes TAPTN a economy and environment friendly method for large graphs.

The guidance mechanism is only effective for the second iteration of TAPTN on this dataset, probably because LLMs only show obvious inconsistency when it use 2-order neighborhood, and reasoning process of small LLMs can't be used as good examplars. However, without two improvements of TAPTN, incorporating 2-order neighborhood can't bring any benefit. See Appendix A for detailed time complexity analysis.

4 Are LLMs' Ability of Leveraging Graph Structure Information on Par with GNNs?

Sections 2 and 3 have shown that general-purpose LLMs are structurally sensitive: when we perturb edges while keeping node texts fixed on low-homophily graphs, accuracy drops systematically, especially for stronger models. Moreover, TAPTN converts this latent sensitivity into reliable gains by performing edge-aware, iterative aggregation in text space. We now ask a more stringent question:

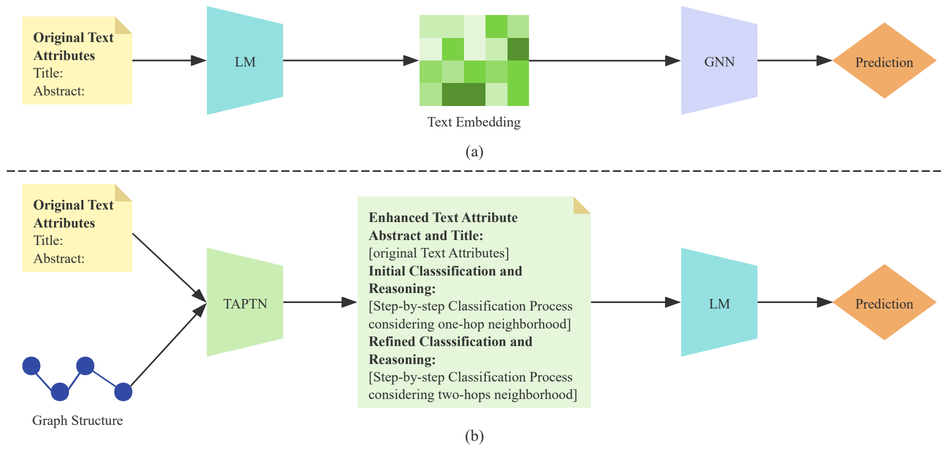

Original Text Attributes
Title:
Abstract:
LM
Text Embedding
GNN
Prediction
(a)
Original Text Attributes
Title:
Abstract:
Graph Structure
TAPTN
Enhanced Text Attribute
Abstract and Title:
[original Text Attributes]
Initial Classsification and Reasoning:
[Step-by-step Classification Process considering one-hop neighborhood]
Refined Classsification and Reasoning:
[Step-by-step Classification Process considering two-hops neighborhood]
LM
Prediction
(b)

Figure 6: Subgraph (a) shows the pipeline of training GNNs on text embeddings of original text attributes. Subgraph (b) shows the pipeline of fine-tuning LMs on Enhanced Text Attributes generated by TAPTN.

When provided with a reasonably designed ICL architecture such as TAPTN, is the structural utilization capability of LLMs still inferior to state-of-the-art GNNs on text-attributed graphs?

The performance gap likely arises from systematic biases between the LLMs' pre-training data and downstream dataset labeling standards, rather than TAPTN's capacity to utilize graph structural information or unreasonable classification. Therefore, TAPTN remains a viable alternative to GNNs.

Answering this requires a comparison where (i) both sides have access to the same node text encoder, (ii) both operate over the same edge structure, and (iii) any remaining performance gap can be attributed to differences in how they use graph structure, not to differences in raw textual capacity or training data.

To support this, we fine-tuned a small-scale language model (LM) using TAPTN-generated enhanced text attributes to mitigate potential biases. In the control group, we fine-tuned the LM using only the nodes' original text attributes (TA) and then used the LM-generated embeddings to train various GNNs. Detailed pipelines for both methods are shown in Figure 6.

4.1 Experiment Setup: Aligning TAPTN and GNN

We consider two moderately homophilic citation graphs with standardized GNN baselines: **Cora** and **Arxiv-2023**. For both datasets, we align the LLM and GNN pipelines around a shared text encoder:

- **TAPTN+LM.** As shown in Figure 6(a), we first run 2-hop TAPTN with DeBERTa as the backend LLM to generate an enhanced text attribute $T_i^{(2)}$ for each node, using the edge-aware templates and instructions from Section 3. We then fine-tune DeBERTa as a standard text classifier on these enhanced texts, using the node labels as supervision. Importantly, during fine-tuning the LM sees no neighborhood structure—all structural information must be encoded into $T_i^{(2)}$ by TAPTN's ICL stage.
- **TA+GNN+LM (control).** As shown in Figure 6(b), in the control pipeline, we fine-tune the same DeBERTa-base LM using only the original text attributes (TA) of nodes, again without any graph structure. We then freeze this LM and use it to generate node embeddings, which feed into various GNNs trained on the original graph. Thus, structural information is exploited only by the GNN (via the adjacency matrix), not by the LM.
- **TA+LM and GraphICL+LM.** We also include TA+LM (LM classifier on original texts only, no graph) and GraphICL+LM (GraphICL-style neighborhood descriptions fed directly to the LM, then

Table 9: Node classification accuracy of TAPTN+LM versus various GNNs trained on TA embeddings across Cora and Arxiv-2023 datasets. All methods include "+LM" by default. The p-value is from a paired t-test. TAPTN+LM significantly outperforms TA+LM by +5.37% ($p < 0.01$), GraphICL+LM by +1.66% ($p < 0.1$), and rivals SOTA GNNs, even surpassing many widely used GNNs.

Method	Cora	Arxiv-2023	p-value
Deberta trained on raw TA embeddings			
TA	77.54±2.38	89.59±0.62	2.314e-4
Normal GNNs (trained on TA embeddings)			
TA+GAT	84.92±1.82	90.56±1.20	2.426e-1
TA+SAGE	84.73±1.44	90.66±0.88	1.630e-1
TA+GATv2	83.95±1.48	90.94±0.26	6.863e-2
TA+GCNII	78.60±2.16	90.94±1.59	4.189e-3
SOTA GNNs (trained on TA embeddings)			
TA+ASDGN	78.23±2.52	90.94±2.17	4.708e-3
TA+RevGAT	84.87±1.54	91.91±1.45	6.113e-1
TA+DirGNN	85.61±1.81	91.52±1.37	8.669e-1
TA+ACMGNN	81.24±2.30	91.62±0.47	2.087e-2
TA+DMP	33.35±2.48	92.39±0.76	2.603e-2
TA+FSGNN	85.98±1.68	92.10±0.72	8.371e-1
TA+GraphSaint	84.82±1.81	89.88±0.47	1.303e-1
TA+DGI	85.10±1.06	93.35±0.24	7.846e-1
TA+APPNP	**88.89±1.11**	92.00±1.06	2.681e-1
TA+ChebNet	84.82±1.85	89.60±1.59	1.042e-1
GraphICL+LM (DeBERTa fine-tuned on GraphICL-enhanced text)			
GraphICL	83.21±1.96	92.29±1.60	6.550e-2
TAPTN+LM (DeBERTa fine-tuned on TAPTN-enhanced text)			
TAPTN	85.15±0.70	**93.64±1.25**	/

fine-tuned) as baselines, to isolate the contribution of TAPTN's structured utilization from both "no structure" and "unstructured structure" controls.

To ensure that TAPTN+LM and TA+GNN+LM see neighborhoods of comparable depth, we use 2-hop TAPTN and set the number of graph convolution layers in each GNN to 2. We conducted experiments on the Cora (Yang et al., 2016) and Arxiv-2023 (Huang et al., 2024) datasets using Deberta-base as the LM. We evaluated 4 widely used GNN architectures (GAT (Velickovic et al., 2017), GraphSAGE (Hamilton et al., 2017), ChebNet (Defferrard et al., 2016), and DGI (Velickovic et al., 2019)) and 10 SOTA GNNs (GATv2 (Brody et al., 2022), GCNII (Chen et al., 2020), RevGAT (Li et al., 2021), ASDGN (Gravina et al., 2023), DirGNN (Rossi et al., 2024), ACM-GNN (Luan et al., 2022), DMP (Yang et al., 2021), Graph-saint (Zeng et al., 2019), FSGNN (Maurya et al., 2022), and APPNP (Gasteiger et al., 2018)).

During LM fine-tuning, no structural information from the neighborhood was provided. Thus, TAPTN+LM can match the classification performance of TA+GNN+LM only if TAPTN's ability to leverage neighborhood structural information is comparable to that of GNNs.

4.2 Results

Table 9 shows that TAPTN+LM significantly outperformed TA+LM by an average of +5.37% ($p < 0.01$) and GraphICL+LM by +1.66% ($p < 0.1$) across both datasets. Additionally, TAPTN+LM achieved accuracy comparable to SOTA GNN models like RevGAT, DirGNN, DGI, APPNP, and FSGNN ($p > 0.1$). It also significantly outperformed GAT, GCNII, GATv2, ASDGN, ACM-GNN, and DMP by

$+1.25\%, +4.07\%, +1.54\%, +4.40\%, +2.88\%$, and $+30.13\%$, respectively ($p < 0.1$). TAPTN+LM also outperformed SAGE, Graphsaint, and ChebNet by $+1.29\%, +1.80\%$, and $+1.80\%$, respectively, though not significantly. These results indicate that TAPTN's ability to utilize graph structural information is on par with SOTA GNNs and superior to some widely used GNN architectures and GraphICL.

Because all GNN baselines and TAPTN+LM: (i) use the same DeBERTa-base TA encoder, (ii) train on the same labels and graph splits, and (iii) have access to the same edge structure, These gains cannot be explained by more LM parameters or more supervision—both pipelines fine-tune the same LM on the same label sets. The only difference is that TAPTN+LM fine-tunes on structurally enriched texts produced by an explicit structure-aware ICL architecture, whereas TA+LM and GraphICL+LM fine-tune on structure-agnostic or unstructured neighborhood text. In other words, TAPTN's ICL phase successfully converts edge structure (including edge directions and local motifs) into textual representations that make downstream supervised learning easier, even for a relatively small LM. Once equipped with TAPTN-generated structural representations, the LLM's ability to exploit graph structure is on par with that of SOTA GNNs. The performance gap is no longer about "LLMs cannot use edges," but about differences at the level of optimization, parameterization, and inductive bias—all of which TAPTN+LM handles surprisingly well despite being built on an ICL-style textual interface.

4.3 Complementary Evidence from Heterophilic graphs

The comparisons in Table 9 are conducted on homophilic citation graphs. To test whether TAPTN-based LLMs remain structurally competitive when homophily is low and potentially misleading, we further evaluate on three malignant heterophilic WebKB datasets (Texas, Wisconsin, Cornell), whose homophily falls in or near the "mid-homophily pitfall" regime (roughly $(0.1, 0.3)$) where many standard GNNs either fail to benefit from neighborhood aggregation or degrade due to heterophilic mixing.

We selected 3 SOTA models designed to addressing heterophilic graphs (ACM-GNN (Luan et al., 2022), DMP (Yang et al., 2021) and FSGNN (Maurya et al., 2022) as baselines. For normal GNNs, we selected 5 widely used or SOTA models (GAT (Velickovic et al., 2017), ChebNet (Defferrard et al., 2016), GATv2 (Brody et al., 2022), Graphsaint (Zeng et al., 2019), APPNP (Gasteiger et al., 2018)).

Table 10 reports performance on the three malignant heterophilic datasets. TAPTN+LM achieves 96.13% average accuracy, exceeding both FSGNN and ACM-GNN by +2.74% on average and outperforming DMP by +49.87%. This supports our thesis: once equipped with a reasonably designed ICL architecture, LLM-based predictors can match or surpass specialized GNNs even in the most challenging heterophilic regime.

5 Related Works

ICL-based methods have proven effective in tasks such as Knowledge Graph Question Answering (KGQA) and topological structure understanding. For example, Guo et al. (2023) demonstrated that LLMs can extract basic structural information from adjacency lists, but not whether LLMs can exploit it. Sun et al. (2023) showed LLMs performing beam search on knowledge graphs to achieve SOTA KGQA, yet this involved selecting relevant triplets rather than applying graph structure.

Node classification tasks demand effective extraction and utilization of graph structural information through ICL. While Guo et al. (2023) introduced ICL to node classification and found neighborhood information enhancing performance, their accuracy did not exceed 60%. Similarly, Hu et al. (2023) reported only slight enhancements or even negative effects when incorporating neighborhood information, with GPT-3.5's accuracy on OGBN-Arxiv and Cora not exceeding 65%.

Research on ICL-based node classification has primarily focused on prompt construction, with limited investigation into effectively leveraging graph structural information. Existing methods often depend on high-quality labeling or underperform compared to GNNs. Notably, there are no practical zero-shot ICL-based node classification methods. Chen et al. (2024) found that adding neighborhood information improves accuracy due to homophily but did not clarify whether it was interpreted as graph structure or linear text. Das et al. (2023) compared different modalities, finding text to be most effective, yet the study relied on

Table 10: Node classification accuracy (%) on malignant heterophilic WebKB graphs (Texas, Wisconsin, Cornell). We report representative normal GNNs+LM, heterophily-specialized GNNs+LM, and TAPTN+LM. Avg is the mean over the three datasets.

Method	**Texas**	**Wisconsin**	**Cornell**	**Avg.**
Normal GNNs (trained on TA embeddings)				
GAT	71.15	61.54	58.00	63.56
GATv2	73.08	80.00	82.00	78.36
APPNP	80.77	72.31	80.00	77.69
GraphSaint	88.46	78.64	80.00	82.37
ChebNet	78.85	95.38	88.00	87.41
Heterophily-specialized GNNs (trained on TA embeddings)				
DMP	38.46	52.31	48.00	46.26
FSGNN	92.31	93.85	94.00	93.39
ACM-GNN	92.31	93.85	94.00	93.39
TAPTN+LM (DeBERTa fine-tuned on TAPTN-enhanced text)				
TAPTN+LM	98.08	92.31	98.00	96.13

high-quality labels and did not assess additional benefits from graph structure narratives. Wang et al. (2023) proposed a Retrieval-Augmented Generation (RAG) strategy that outperformed GNNs by creating few-shot exemplars from similar training samples, but it heavily depended on high-quality annotations, lacking zero-shot learning capabilities.

As for exploration of LLMs' ability to leverage graph structural information through ICL, Huang et al. (2024) concluding that LLMs treat neighborhood information merely as linear text through rewiring experiments. GraphICL (Sun et al., 2025) systematizes this direction by proposing a unified ICL benchmark and prompt template for graph reasoning tasks, combining anchor-node text, task description, structure-aware neighbor texts, and labeled demonstrations. Importantly, its ablations highlight that selecting homogeneous neighbors/demonstrations (e.g., most similar or class-aligned) often contributes the largest portion of performance gains, which can reinforce the interpretation that "structure helps mainly as a retrieval mechanism for homophilic texts". However, their methodology was flawed: the absence of performance changes after removing structural descriptions might reflect high homophily rather than an inability to leverage graph structure. The prompt templates they use do not encode edge semantics, and the ICL framework they employ lacks a dedicated mechanism for handling lengthy multi-hop neighborhood contexts and performing graph-structure reasoning. Xu et al. (2025) points out that when LLMs are used as the backbone network, the structural signals encoded by GNNs do not provide any benefit and may even be detrimental. However, they overlooked exploring whether LLMs can understand and utilize the structural information of text attribute graphs within the text space through a specific ICL framework.

6 Conclusion

This work revisits the central question: **Can general-purpose LLMs genuinely leverage graph structural information in text-attributed graphs via ICL, once we remove confounding factors and endow them with an architecture explicitly designed for structural reasoning?** Our answer is affirmative.

First, by moving away from high-homophily citation graphs and instead operating on low-homophily WebKB datasets with carefully controlled rewiring, we demonstrate that LLMs are inherently sensitive to graph structure. Flipping first-order edge directions produces consistent and often substantial accuracy drops across seven diverse LLMs, with structural sensitivity strongly correlated with model capability. Extreme two-hop rewiring yields smaller but still universal degradation, with sensitivity more uniform across models due to the coarse, count-based representation of second-order neighborhoods. In all cases, node texts and label distributions are held fixed, so performance differences can only be explained by changes in how the models interpret connectivity patterns, not by differences in homophilic neighbor texts. The main barriers to harnessing this latent structural ability are architectural rather than inherent: traditional GraphICL prompts linearize large ego-networks, lacking a dedicated mechanism for graph structure reasoning and handling lengthy multi-hop neighborhood contexts, and suffering from inconsistent adherence to correct graph structure utilization method.

Second, we show that TAPTN addresses these issues by (i) encoding first-order neighborhoods with a structure-aware template that foregrounds edge directions and roles, (ii) iteratively passing and summarizing textual "messages" to build multi-hop representations, and (iii) stabilizing reasoning with self-generated step-by-step instructions at each aggregation and classification step. Experiments on five TAG benchmarks confirm that TAPTN consistently and substantially improves over zero-hop CoT and GraphICL-style baselines, with gains growing when moving from 1-hop to 2-hop neighborhoods and being especially pronounced on malignant heterophilic graphs, where homophily-based heuristics are ineffective. Ablation studies further show that naive self-reflection cannot reproduce these gains, whereas structured aggregation and instructions together provide stable benefits and convert structural sensitivity into reliable accuracy improvements.

Third, by aligning LLM and GNN pipelines around the same DeBERTa text encoder, we show that TAPTN's structural utilization capability is competitive with state-of-the-art GNNs. Fine-tuning an LM on TAPTN-enhanced texts (TAPTN+LM) yields performance that matches or exceeds many widely used architectures (e.g., GAT, SAGE, ChebNet) and is statistically indistinguishable from several strong SOTA models on homophilic citation graphs, while clearly outperforming both normal and heterophily-specialized GNNs on malignant WebKB datasets. Since all methods share the same encoder, labels, and edge structure, the remaining gap cannot be attributed to lacking access to structural information; instead, TAPTN+LM demonstrates that an ICL-style textual interface is sufficient to exploit graph structure information, including edge semantics and local topology as effectively as dedicated graph neural networks.

Taken together, these results overturn the view that LLMs "cannot use graph structure" in TAG node classification. Rather, prior negative findings largely stem from homophily-centric datasets and prompt designs that obscure structural cues. Once these confounders are removed and an architecture like TAPTN is introduced to expose and guide structural reasoning, general-purpose LLMs not only exhibit clear structural sensitivity but also translate it into competitive downstream performance. This suggests a practical path forward: instead of replacing graph models outright, we can embed graph structural reasoning into LLM-centric workflows, using TAPTN-style ICL as a zero-shot or low-supervision alternative when labels are scarce, and as a complementary structural encoder when integrating with traditional GNNs. Future work includes extending TAPTN beyond node classification to link prediction and subgraph-level tasks, learning or optimizing instruction policies, and further reducing computational cost for very large graphs, thereby broadening the range of applications where LLMs can serve as structure-aware graph learners.

References

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=F72ximsx7C1`.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International conference on machine learning*, pp. 1725–1735. PMLR, 2020.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (llms)in learning on graphs. *SIGKDD Explor. Newsl.*, 25(2):42–61, mar 2024. ISSN 1931-0145. URL `https://doi.org/10.1145/3655103.3655110`.

Debarati Das, Ishaan Gupta, Jaideep Srivastava, and Dongyeop Kang. Which modality should i use–text, motif, or image?: Understanding graphs with large language models. *arXiv preprint arXiv:2311.09862*, 2023.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

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

Rayid Ghani, Rosie Jones, Andrew McCallum, Tom Mitchell, Dunja Mladenic, Kamal Nigam, and Sean Slattery. Cmu world wide knowledge base project. https://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/, 2001. Accessed: 2024-08-08.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1263–1272. JMLR.org, 2017.

Alessio Gravina, Davide Bacciu, and Claudio Gallicchio. Anti-Symmetric DGN: a stable architecture for Deep Graph Networks. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=J3Y7cgZOOS.

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

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *arXiv preprint arXiv:2305.19523*, 2023.

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

Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. Can LLMs effectively leverage graph structural information through prompts, and why? *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=L2jRavXRxs.

Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational complexity of self-attention. In *International Conference on Algorithmic Learning Theory*, pp. 597–619. PMLR, 2023.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pp. 6437–6449. PMLR, 2021.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35:1362–1375, 2022.

Sitao Luan, Chenqing Hua, Qincheng Lu, Liheng Ma, Lirong Wu, Xinyu Wang, Minkai Xu, Xiao-Wen Chang, Doina Precup, Rex Ying, et al. The heterophilic graph learning handbook: Benchmarks, models, theoretical analysis, applications and challenges. *arXiv preprint arXiv:2407.09618*, 2024a.

Sitao Luan, Chenqing Hua, Minkai Xu, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Jie Fu, Jure Leskovec, and Doina Precup. When do graph neural networks help with node classification? investigating the homophily principle on node distinguishability. *Advances in Neural Information Processing Systems*, 36, 2024b.

Tailong Luo and Dingyuan Zhang. Research on financial credit fraud detection methods based on temporal behavioral features and transaction network topology. *Artificial Intelligence and Machine Learning Review*, 5(1):8–26, 2024.

Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Simplifying approach to node classification in graph neural networks. *Journal of Computational Science*, 62:101695, 2022.

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

Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. Disentangled representation learning with large language models for text-attributed graphs. *arXiv preprint arXiv:2310.18152*, 2023.

Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M Bronstein. Edge directionality improves learning on heterophilic graphs. In *Learning on Graphs Conference*, pp. 25–1. PMLR, 2024.

T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv preprint arXiv:2303.10993*, 2023.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model with knowledge graph. *arXiv preprint arXiv:2307.07697*, 2023.

Yuanfu Sun, Zhengnan Ma, Yi Fang, Jing Ma, and Qiaoyu Tan. GraphICL: Unlocking graph learning potential in LLMs through structured prompt design. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2440–2459, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 491–500, New York, NY, USA, 2024. Association for Computing Machinery. URL `https://doi.org/10.1145/3626772.3657775`.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, N. Chawla, and Panpan Xu. Graph neural prompting with large language models. In *AAAI Conference on Artificial Intelligence*, 2023. URL `https://api.semanticscholar.org/CorpusID:263152125`.

Guoxiang Tong and Jieyu Shen. Financial transaction fraud detector based on imbalance learning and graph neural network. *Applied Soft Computing*, 149:110984, 2023. URL `https://www.sciencedirect.com/science/article/pii/S1568494623010025`.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

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

Duo Wang, Yuan Zuo, Fengzhi Li, and Junjie Wu. Llms as zero-shot graph learners: alignment of gnn representations with llm token embeddings. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.

Qinyong Wang, Zhenxiang Gao, and Rong Xu. Graph agent: Explicit reasoning agent for graphs. *arXiv preprint arXiv:2310.16421*, 2023.

Sizheng Wei and Suan Lee. Internet fraud transaction detection based on temporal-aware heterogeneous graph oversampling and attention fusion network. *PloS one*, 20(12):e0337208, 2025.

Xixi Wu, Yifei Shen, Fangzhou Ge, Caihua Shan, Yizhu Jiao, Xiangguo Sun, and Hong Cheng. When do llms help with node classification? a comprehensive analysis. *arXiv preprint arXiv:2502.00829*, 2025.

Haotian Xu, Yuning You, and Tengfei Ma. When structure doesn't help: Llms do not read text-attributed graphs as effectively as we expected. *arXiv preprint arXiv:2511.16767*, 2025.

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD workshop on mining data semantics*, pp. 1–8, 2012.

Liang Yang, Mengzhe Li, Liyang Liu, Chuan Wang, Xiaochun Cao, Yuanfang Guo, et al. Diverse message passing for attribute with heterophily. *Advances in Neural Information Processing Systems*, 34:4751–4763, 2021.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 40–48. JMLR.org, 2016.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pp. 1955–1973, 2024.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Yusheng Zhao, Qixin Zhang, Xiao Luo, Weizhi Zhang, Zhiping Xiao, Wei Ju, Philip S Yu, and Ming Zhang. Dynamic bundling with large language models for zero-shot inference on text-attributed graphs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

A Time Complexity Analysis and Experiments

TAPTN's complexity depends on LLM operations, where output sequence of a node includes LLM-generated categorization and reasons for it while input sequence of a node in the (l+1)-th iteration consists of enhanced text attributes of its first-order neighbors (including original attributes and LLM-outputs of the l-th iteration). For a specific graph, the enhanced text attributes and outputs length of a node float around fixed values p, q respectively. According to Keles et al. (2023), time complexity for a self-attention based model to output a sequence of m tokens is:

$$T_m = O(d \cdot m \cdot n^2), \tag{11}$$

where d is the number of self-attention layers, which is a constant for a specific model.

Thus, for a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}, \mathbf{E}$ is nodes and edges set of $\mathbf{G}$ respectively, TAPTN's per-node complexity is:

$$O(m \cdot n^2) = O(q \cdot (\frac{|\mathbf{E}|}{|\mathbf{V}|}p + q)^2). \tag{12}$$

As there are $|\mathbf{V}|$ nodes in G, the time complexity for a single iteration is:

$$O(|\mathbf{V}| \cdot q \cdot (\frac{|\mathbf{E}|}{|\mathbf{V}|}p + q)^2) = O(\frac{|\mathbf{E}|^2}{|\mathbf{V}|} + |\mathbf{E}|). \tag{13}$$

For L iterations, it becomes $O((\frac{|\mathbf{E}|^2}{|\mathbf{V}|} + |\mathbf{E}|) \cdot L)$. For dense graphs, this scales as $O(|\mathbf{V}|^3 \cdot L)$, manageable for small-to-medium graphs, and for large sparse graphs which more common in real applications, it simplifies to $O(|\mathbf{V}| \cdot L)$. Additionally, as the context window of a specific LLM is limited, when a graph is become so dense that first-order neighborhood descriptions of an average node exceed context window, only a constant number C neighbors will be chosen. In this case, $|\mathbf{E}| = C|\mathbf{V}|$, so its time complexity also simplifies to $O(|\mathbf{V}| \cdot L)$. In conclusion, the time complexity of TAPTN is $O(|\mathbf{V}| \cdot L)$, making TAPTN a efficient zero-shot learning algorithm.

B Prompts Templates, Neighborhood Description Templates, Task Descriptions and Step-by-step Instruction

B.1 Prompts Templates

Prompts Templates

[Task Description Head]
[Step-by-step Instruction]
[Neighborhood Description]
[Task Description Ending].

B.2 Neighborhood Description Templates

The structure-aware neighborhood description template for Cora and Arxiv-2023 is listed as below:

Structure-Aware Template for Citation Graphs

#### Paper ####
[content description of the target paper]
#### Citations ####
The paper has following citations:
[content description of each citation]
#### References ####
The paper has following refernces:
[content description of each reference]
, where the content of a paper is described as below:
### [paper Number] ###
## Abstract ##
[]abstract]
## Title ##
[title]
Citations Number: [citations number]
References Number: [references number]
## Initial categorization ##
[initial categorization]
## Reasons for initial categorization ##
[reasons for initial categorization]
, where paper number describes the relation between the described paper and the target paper (for example, Citation 1). For the first iteration, there won't be the "initial categorization" and "reasons for initial categorization" parts.

The neighborhood description template for Texas, Wisconsin, Washington and Cornell for rewiring experiment in Section 2 is listed as below:

Neighborhood Description Template for WebKB Graphs (Rewiring Experiments)

#### Webpage ####
## Incoming links number ##
[]incoming links number]
## Outgoing links number ##
[outgoing links number]
URL: [URL]

#### It has inbound links from following webpages: ####
[description of inbound link]
#### It has outbound links to following webpages: ####
where a linked webpage is described as below:
### [Webpage Number] ###
It has outbound link to webpage [URL], which is a []category] page with content abstract as below: [content abstract], with link pattern as belows: [link pattern]
, where the content abstract part is only added when its category is "other", and the link pattern part is described as below:
Inbound links by category:
[category]: [number of second-order neighbors linking to this webpage belonging to each category]
Outbound links by category:
[category]: [number of second-order neighbors linked from this webpage belonging to each category].

The target page itself is not involved in these link pattern. If a first-order page has no hyperlinks except the target page, its link pattern is recorded as "private resource".

The structure-aware neighborhood description template for Texas, Wisconsin, Washington and Cornell for TAPTN in Section 3 is listed as below:

Structure-Aware Template for Citation Graphs (TAPTN)

## Incoming links number ##
[]incoming links number]
## Outgoing links number ##
[outgoing links number]
URL: [URL]
Content Abstract: [content abstract of the target webpage]
## Initial categorization ##
[initial categorization]
## Reasons for initial categorization ##
[reasons for initial categorization]
#### It has inbound links from following webpages: ####
[description of inbound link]
#### It has outbound links to following webpages: ####
The paper has following refernces:
[description of each outbound link]
where a linked webpage is described as below:
It has inbound/outbound link from/to webpage [URL] with content abstract as below: [content abstract].
## Initial categorization ##
[initial categorization]
## Reasons for initial categorization ##
[reasons for initial categorization].

For the first iteration, there won't be the "initial categorization" and "reasons for initial categorization" parts.

B.3 Task descriptions

Task descriptions used for Cora are listed as below:

Task Description Head for the First Iteration (Cora)

Please predict the 2 most appropriate categories for the paper based on the titles and abstracts of the paper itself as well as its references and citations. Choose from the following categories: [candidate options]

Task Description Head for Other Iteration (Cora)

Further revise the initial categorization for the paper. Choose the most appropriate category for the paper from the following categories:[candidate options]Now, apply this method to the following paper. If multiple options apply, ensure these options are sorted from the most relevant to the least relevant.

Task description ending (Cora)

####Question####: Predict the 2 most appropriate category for the paper. For each category you predict, give a relevance score from 0 and 1. Make double choices from the given list of categories: [candidate options]. Answer: Let's think step by step.

Task Description for Parser (Cora)

Please extract the most appropriate category for the paper. Make single choise from the following categories: [candidate options]. Answer:

Task descriptions used for Arxiv-2023 are listed as below:

Task Description Head for the First Iteration (Arxiv-2023)

Predict the 2 most appropriate arXiv Computer Science (CS) sub-category for the paper based on the titles and abstracts of the paper itself as well as its references and citations. Choose from the following categories: [candidate options]. The predicted sub-category should be in the format 'cs.XX'.

Task Description Head for Other Iteration (Arxiv-2023)

Further revise the initial categorization for the paper. Predict the 2 most appropriate arXiv Computer Science (CS) sub-category for the paper. The predicted sub-category should be in the format 'cs.XX'. Now, apply this method to the following paper. The predicted sub-category should be in the format 'cs.XX'. If multiple options apply, ensure these options are sorted from the most relevant to the least relevant.

Task description ending (Arxiv-2023)

####Question####: Predict the 2 most appropriate category for the paper. For each category you predict, give a relevance score from 0 and 1. Make double choices from the given list of categories: [candidate options]. Answer: Let's think step by step.

Task Description for Parser (Arxiv-2023)

Please extract the refined most appropriate category for the paper. Choose from the following categories: [candidate options]. Answer:

Task descriptions used for Texas, Wisconsin, Washington and Cornell are listed as below:

Task Description Head for Rewiring Experiments (WebKB)

Please predict the 2 most appropriate categories for the webpage based on the URL and category of the webpages which the target page has outbound links to or has inbound linked from (for the non-main-page or resource pages labeled as 'other' within these webpages, their content abstract will be attached in addition) as well as the link pattern (inbound and outbound hyperlinks) of the target webpage. Choose from the following categories: [candidate options].

Task Description Head for TAPTN (WebKB)

Please predict the 2 most appropriate categories for the webpage based on the URL and content abstract of the webpage as well as its linked pages which the target page has outbound links to or has inbound links from. Choose from the following categories: [candidate options].

Task description ending (WebKB)

####Question####: Predict the 2 most appropriate categories for the webpage with URL: . For each category you predict, give a relevance score from 0 and 1. Make double choices from the given list of categories: [candidate options]. Answer: Let's think step by step.

Task Description for Parser (WebKB)

Please extract the category with highest relevance score for the webpage. Make single choise from the following categories: [candidate options]. Answer:

B.4 Step-by-step instructions

The step-by-step instruction used for Cora and Arxiv-2023 in the first iteration is listed as below:

Step-by-step Instruction for the First Iteration (Citation Graphs)

Choosing the most appropriate category for a paper based on the titles and abstracts of the paper and its references and citations involves a process of identifying key themes, methods, and subject areas. Here is a structured approach to help you categorize a paper:
1. **Understand the Categories**: First, familiarize yourself with the predefined list of categories available. Understand what each category entails, including the typical methodologies, subject areas, and scopes covered by each.
2. **Analyze the Paper's Abstract and Title**:
- **Keywords and Phrases**: Identify key terms and phrases in the title and abstract. These often indicate the central themes and the discipline.
- **Objective and Approach**: Look for statements about the paper's main objectives and the methods used. This can give clues about whether the paper is theoretical, empirical, review-based, etc.
- **Subject Area**: Determine which area of study the paper belongs to based on the problems addressed and the context provided.
3. **Examine References**:
- **Reference Sources**: Review the titles and abstracts of the cited papers. Papers often cite sources within the same or a closely related field.
- **Patterns in Citations**: Note any recurring themes or predominant disciplines in the citations. This can indicate the community and academic discourse the paper is engaging with.
4. **Check Citations to the Paper**:
- **Citing Papers' Focus**: Look at what aspects of the paper are being cited by other authors and in what context. This might provide insights into the paper's contributions and relevance in specific

fields.
5. **Synthesize the Information**:
- **Majority Rule**: If the majority of references and citations belong to a particular category, there's a good chance the paper fits there too.
- **Consistency Check**: Ensure the identified category aligns with the paper's objectives and methods.
- **Broader Context**: Consider if the paper might be interdisciplinary and whether it should be categorized under a more general or specific field based on its breadth and focus.
6. **Make a Decision**:
- **Best Fit**: Choose the category that best captures the essence of the paper based on the synthesis of the above steps.
- **Fallback Option**: If unsure, lean towards a broader category that encompasses multiple aspects of the paper.
7. **Document the Reasoning**: It's useful to keep notes on why you categorized the paper in a certain way, especially if the decision was not straightforward. This helps in maintaining consistency when categorizing other papers.
8. **Final Check**: After final decision, review the reasoning process and final categorization carefully and identify any factual errors, inconsistencies, or missing important information. If you find any issue, please fix it accordingly to ensure it logically fits with its content and its scholarly context. This final check ensures that the category reflects the paper's contributions and themes accurately.
This method relies heavily on critical thinking and a good grasp of the subject areas represented in your category list. It requires an analytical approach to text and the ability to discern patterns and themes from limited information.

The step-by-step instruction used for Cora and Arxiv-2023 in other iterations is listed as below:

Step-by-step Instruction for Other Iteration (Citation Graphs)

To further revise the categorization of a paper based on the citation network and the initial categorizations, follow this structured method:
1. **Examine the Citation Network**:
- **Analyze Connections**: Look at how the paper is connected within the citation network. Identify whether it is primarily citing or being cited by papers within specific categories.
- **Identify Influential Papers**: Determine which papers in the citation network are highly influential or frequently cited. These papers can often guide you towards the core category of the subject matter.
2. **Compare Initial Categorizations**:
- **Consistency Check**: Check if the initial categorizations of the papers within the citation network align with the initial categorization of the target paper. A strong alignment suggests a correct initial categorization.
- **Majority Rule**: If the majority of the papers in the citation network belong to a particular category, this might indicate the central focus area for the target paper.
3. **Review Reasoning for Categorizations**:
- **Justifications**: Evaluate the reasons given for the initial categorizations of the papers in the citation network. Strong, well-articulated justifications can help validate the categories.
- **Identify Common Themes**: Look for common themes in the reasoning provided. If similar reasons are repeatedly used for categorizing papers into a specific category, this strengthens the case for that category.
4. **Cross-Referencing Themes**:
- **Abstract and Title Analysis**: Re-examine the abstracts and titles of the target paper and the papers in its citation network. Look for shared keywords, phrases, and thematic overlaps.
- **Methodologies and Approaches**: Compare the methodologies and approaches described in the abstracts. Similar methods often indicate similar categorical alignment.

5. **Consider the Influence of Interdisciplinary Connections**:
- **Broader Context**: Determine if the paper spans multiple disciplines. If it does, consider which categories are most relevant based on the depth and focus of the interdisciplinary connections.
- **Primary vs. Secondary Categories**: If the paper is highly interdisciplinary, you might need to choose a primary category that best represents the core contribution and a secondary category for the supporting discipline.
6. **Iterative Adjustment**:
- **Re-Evaluate Initial Judgment**: Based on the analysis of the citation network and the comparisons made, re-evaluate the initial categorization.
- **Revise if Necessary**: Adjust the category if the evidence from the citation network strongly supports a different categorization.
7. **Final Decision**:
- **Best Fit Category**: Choose the category that now seems to best capture the essence of the paper, considering the additional information from the citation network.
- **Document the Revision**: Make notes on why the revision was made, including the influence of the citation network and any key papers that led to the change in categorization.
8. **Final Review**:
- **Self-Check**: Review the analysis process and the final categorization decision carefully and identify any factual errors, inconsistencies, or missing important information. If you find any issue, please fix it accordingly to ensure that the analysis process is logically correct and fitting with the paper itself as well as its citation network, the final categorization decision aligns with the overall analysis and that the paper is placed where it best fits within the academic landscape.
By systematically analyzing the citation network and considering the broader context provided by the initial categorizations and reasons, you can refine and improve the judgment of the most appropriate categories for the paper. This approach ensures that the categorization is robust, well-justified, and reflective of the paper's true academic context.

The step-by-step instruction used for Texas, Wisconsin, Washington and Cornell is listed as below:

Step-by-step Instruction (WebKB Graphs)

To classify these pages manually without the aid of a machine learning model, you can follow a systematic approach by analyzing the available data (link patterns and content abstracts for "other" pages). Here's a step-by-step guide to help you with the classification process:
### Step 1: Analyze Link Patterns
Link patterns can provide significant clues about the category of a webpage. Consider both the inbound and outbound links:
1. **Inbound Links:**
- **From Faculty Pages:** Indicates the target page may be important for faculty members, possibly related to faculty, projects, or research. Somtimes students also have single inbound link from faculty.
- **From Student Pages:** Indicates relevance to students, potentially a course, faculty, department, or project page.
- **From Course Pages:** Likely indicates a relationship with academic courses for example, faculty, student or course materials. - **From Department Pages:** May suggest the page is important for the entire department, possibly an administrative or project page.
- **From Staff Pages:** Could imply relevance to staff-related activities.
- **From Project Pages:** Suggests involvement in specific projects, may be faculty or students.
- **From Other Pages:** Determine according to the content of this "other" page. For example, inbound links from webpages that list information of graduate students (such as student directories) indicate it's a student page, inbound links from course list indicate it's a course page, etc.
2. **Outbound Links:**
- **To Faculty Pages:** The target page could be related to faculty activities or information, for

example, it may be a student page the faculty is supervising, the project page the faculty is leading, or the course page the faculty is teaching, etc.
- **To Student Pages:** The target page might be course, faculty pages, or it providing resources or information useful/related to students such as students directories.
- **To Course Pages:** Likely to be faculty/student pages as teacher/TA, or it is related to academic content or course information.
- **To Department Pages:** The personal mainpages (staff/faculty/student) has outbound links back to department mainpage. It indicates a broader departmental focus if it's an "other" page.
- **To Staff Pages:** Might be providing staff-related resources or information.
- **To Project Pages:** Suggests the target page is project-related, for example, it's a faculty page or student page as a participant.
- **To Other Pages:** Consider the content of the "other" page to infer the target page's category. For example, outbound links to research publications indicate a faculty/student page, outbound links to miscellaneous personal content indicate a student page, outbound links to administrative documents indicate a staff page, outbound to course materials indicate a course page, etc.
### Step 2: Content Abstract Analysis (for "Other" Pages)
For pages labeled as "other" with attached content abstracts, examine the abstracts closely to understand the primary focus of the page. Look for keywords and phrases that indicate:
- **Academic Terminology:** Course names, academic terms, syllabus details, etc., indicate course pages.
- **Research Terminology:** Research interests, publications, projects, etc., suggest faculty or project pages.
- **Administrative Language:** Departmental policies, staff roles, administrative announcements, etc., indicate department or staff pages.
### Step 3: Contextual Linking and Category Inference
Integrate the insights from link patterns and content abstracts to infer the category:
1. **Faculty Pages:**
- High number of inbound links from course, project, students or other faculty pages.
- Outbound links to research interests, publications, projects related content, courses related content or departmental resources. Sometimes to student pages.
2. **Student Pages:**
- Inbound links from course, project or other student pages. Often with an inbound link from webpages that list information of graduate students (such as student directories). Sometimes a single inbound link from faculty page.
- Outbound links to course-related content, project related content, faculty members or miscellaneous personal content. Sometimes with a single outbound link to department mainpage.
3. **Course Pages:**
- Inbound links from student pages or faculty pages.
- Outbound links to faculty members, students, syllabi, and academic resources.
4. **Department Pages:**
- High number of inbound links from all categories (faculty, student, staff, etc.).
- Outbound links to department-wide resources (including content related to research, students, faculty, course, administrative, etc.).
5. **Staff Pages:**
- Inbound links from departmental or faculty pages.
- Outbound links to administrative documents, departmental resources.
6. **Project Pages:**
- Inbound links from faculty and student pages.
- Outbound links to research-related content, publications, participants (faculty/student) and external resources.
7. **Other Pages:**
- Typically have content abstracts attached.

Table 11: The detailed statistics of the datasets used in experiments. Train/Eval/Test splitting is not applicable for Washington and Amazon datasets, as they are not used for LM fine-tuning. For Texas, Wisconsin and Cornell datasets, the three parts splitted by slashes are number of nodes for rewiring experiments in Section 2, ICL-based methods in Section 3 and finetuned LMs in Section 4 respectively.

Datasets	**#Nodes**	**#Edges**	**#Train Nodes**	**#Eval Node**	**#Test Node**
Cornell	800	1625	0/0/149	0/0/50	247/248/49
Texas	788	1731	0/0/154	0/0/51	253/256/51
Washington	1119	2372	/	/	257
Wisconsin	1208	3464	0/0/193	0/0/64	314/321/64
Cora	2708	5429	270	542	542
Arxiv-2023	2389	5153	1124	287	315
Amazon	13482	37522	/	/	400

- Serve as auxiliary pages linked primarily to a main category page (faculty, student, etc.).
### Step 4: Cross-Verification
Verify your initial classification by cross-referencing with multiple link patterns and content abstracts. If a page seems ambiguous, check the link patterns again and reassess based on the most frequent category of linking pages.
### Step 5: **Final Review**:
Review the analysis process and the final category carefully and identify any factual errors, inconsistencies, or missing important information. If you find any issue, please fix it accordingly to ensure that the analysis process and the final category of target page are logically correct and fitting with its context.
By carefully analyzing the link patterns and content abstracts, you can systematically classify the webpages into their respective categories.

C Detailed Description of Datasets Used in Our Research

The detailed descriptions of the datasets used in this paper are as below:

- WebKB dataset: Consisting of Cornell, Texas, Washington and Wisconsin. This data set contains WWW-pages collected from computer science departments of various universities in January 1997 by the World Wide Knowledge Base project of the CMU text learning group. The pages were manually classified into the 7 categories: student, faculty, staff, department, course, project and other. The class other is a collection of pages that were not deemed the "main page" representing an instance of the previous six classes. For example, a particular faculty member may be represented by home page, a publications list, a vitae and several research interests pages. Only the faculty member's home page was placed in the faculty class. The publications list, vitae and research interests pages were all placed in the other category. We just evaluate different methods on the first 6 categories as the intra-class similarities of both hyperlink pattern and content of pages in "other" category are weak, thus they can be categorized as pages in the first 6 categories reasonably. What's more, since most of the pages are labeled as "other", Differences in the classification performance of pages in the first 6 categories will have almost no impact on total accuracy if pages in "other" category are evaluated.

- Cora dataset: The Cora dataset consists of scientific papers categorized into different classes. The primary data structure includes a citation graph where each node represents a paper, and each directed edge represents a citation from one paper to another. The dataset includes both the text content of the papers (which is typically represented as a bag-of-words) and the citation relationships between them. The Cora dataset was originally constructed by McCallum et al. (2000) as part of

Table 12: The detailed architectures of each GNNs. The hidden dims of SAGE and GAT are set to 1024 as their fitting abilities are not as powerful as SOTA GNNs. Number of convolutional layers of all GNNs are set to 2, ensuring the order of neighborhood they consider are the same as TAPTN. "/" means this item is not applicable.

GNN	**Hidden Dim**	**#Layers**	**Special Architecture**
GAT	1024	2	Number of heads is set to 8.
SAGE	1024	2	/
GATv2	128	2	/
GCNII	128	2	/
ASDGN	128	2	We select GAT as its aggregation function. In ASDGN, the hidden dim are the same as input dim, so we first map the inputs to the hidden dim using MLP.
RevGAT	128	2	/
DirGNN	128	2	The basic covolutional layer we select are GCN.
APPNP	128	2	The teleport probability is set to 0.1.
DGI	128	2	We select GCN with hidden dim 128 as the encoder, and mean function as readout function.
ChebNet	128	2	Chebyshev filter size is set to 3.
Graphsaint	128	2	Implemented by GATv2 combining a Graphsaint random walk sampler.
DMP	128	2	/
FSGNN	128	2	/
ACM-GNN	128	2	/

their work on citation indexing and clustering. The papers were collected from the research literature on topics like machine learning, data mining, and other related fields. The papers were manually categorized into 7 classes based on their research topics, specifically Case-Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning and Theory.

- Amazon dataset: This dataset is a node classification dataset that originates from a large Amazon product co-purchasing network. Each node in the graph represents a product, and an edge between two nodes signifies that the two products are frequently purchased together. The dataset includes products categorized into 47 top-level categories on Amazon platform. This dataset was initially introduced by Yang & Leskovec (2012) with primary goal of developing a heuristic parameter-free community detection method that easily scales to networks with more than hundred million nodes. The dataset used in our work is a connected component consist of 13482 nodes.

- Arxiv-2023 dataset: This dataset was constructed by Huang et al. (2024). They sampled 668 test nodes from about 46,000 arXiv CS papers published between January 1 and August 22, 2023, and extracted references to identify one-hop and two-hop neighbors by searching for valid arXiv IDs and titles via the arXiv API. To comply with API rate limits, each paper was limited to

Table 13: Hyper-parameters for training GNNs and fine-tuning LMs. The values in the brackets in the third column are the special setting for training RevGAT on the Cora dataset, where normal hyper-parameters didn't perform well. "/" means this item is not applicable.

Parameter	**LM**	**GNN**
Learning Rate	2.02e-5	0.01 (0.002)
Weight Decay	0.0	0.0
Dropout	0.3	0.0 (0.5)
#Training Epochs	15	200
Batch Size	4	/
Early Stop Patience	/	50
Attention Dropout	0.1	/
Classifier Dropout	0.4	/

30 searches, and unmatched pre-2019 papers were excluded. Using this data, they constructed a citation network, where one-hop references were identified through both arXiv ID matching and title searching, and two-hop references were determined solely by arXiv ID matching. Dataset statistics reveal similar test node degrees between ogbn-arxiv and arxiv-2023. We selected a subgraph consisting of 7 categories for evaluation (specifically, "cs.GT", "cs.MA", "cs.RO", "cs.NE", "cs.IR", "cs.SI", "cs.CY"), then edges that contains nodes that not belong to these categories were deleted, finally we removed nodes with no neighbors. The consideration for the categories selection include that nodes number of these categories are close to each other, and their connotation partially overlap, ensuring a classification method must understand the semantic of every category precisely to achieve a good performance.

The detailed statistics of the datasets we used are presented in Table 11.

D Detailed Setting of Experiments in Section 4

We evaluated LMs fine-tuned on enhanced text attributes generated by TAPTN and other GNN-based models trained on original text attributes for multiple runs (4 runs for Cora and 3 runs for Arxiv-2023). The details of the GNNs' architectures are shown in Table 12.

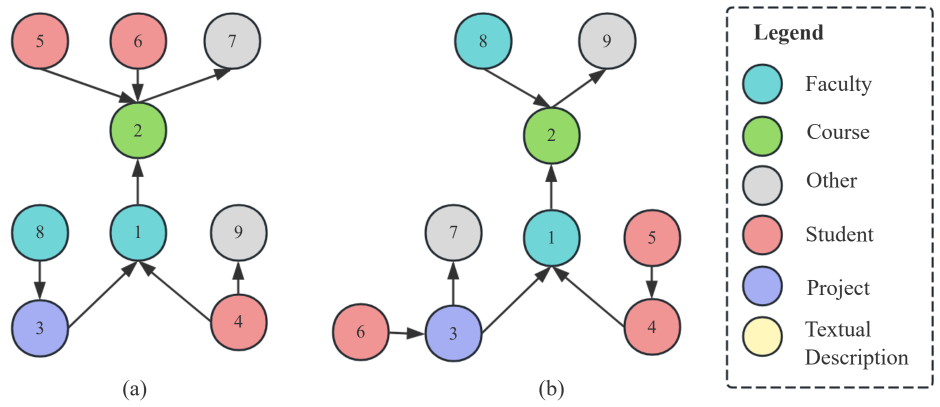
The hyper-parameters for training GNNs and fine-tuning LMs are shown in Table 13. As our training pipelines are modified based on the programs provided by He et al. (2023), their training hyper-parameters are also adopted by us.

E Details of Rewiring Methods Used in Former Researches

"Random" keeps 1-hop neighbors and randomly connects 2-hop neighbors to 1-hop neighbors; "extreme" retains 1-hop neighbors and connects all 2-hop neighbors to a random 1-hop neighbor; "Path" randomly connects all 1-hop neighbors into a path, as shown in figure 7.

F Discussion about Over-smoothing

TAPTN should suffer much less than MPNNs, but we can't prove that. According to (Rusch et al., 2023), the over-smoothing at the n-th layer are measured by the Dirichlet energy, i.e. average Euclidean distance between embeddings of the central nodes and its 1-order neighbors. As TAPTN doesn't follow homophilic hypothesis (similar to ACM-GNN (Luan et al., 2022), which can possibly extract dissimilar neighborhood information), it won't make Dirichlet energy approach 0, meaning it won't over-smoothing frequently. And Transformer has residual connections and normalization layers, which can alleviating over-smoothing (Rusch



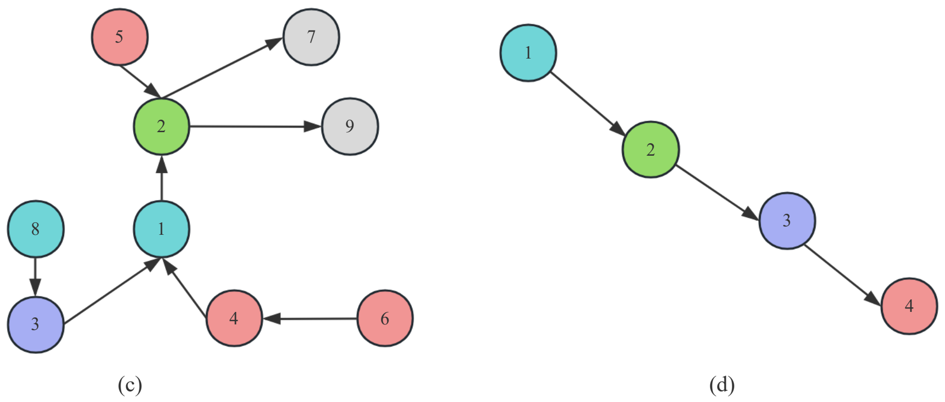


Figure 7: Example of rewiring 2-hop neighborhood for node 1 on texas dataset. “Random” keeps 1-hop neighbors and randomly connect 2-hop neighbors to 1-hop neighbors; “extreme” keeps 1-hop neighbors and connects all 2-hop neighbors connect to a random 1-hop neighbor; “Path” randomly connects 1-hop neighbors as a path.

et al., 2023). However, as a method designed mainly for closed-source LLMs, we can’t access the embeddings to evaluate it.