BEYOND CODE GENERATION OVER TABULAR GRAPHS: A STRUCTURAL CONTEXT TOOL FOR GRAPH INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Models (LLMs) have demonstrated graph inference capabilities on Text-Attributed Graphs (TAGs), which are widely used in social networks and recommendation systems. Graph-as-Code (GAC) enables LLM-Graph interaction in tabular form through code generation, avoiding the need to explicitly serialize large graphs into prompts. In the node classification task, accessibility of tabular graph data does not guarantee that GAC can effectively capture local neighbor structure or global data bias. We observe that the overlap between misclassified nodes of GAC and GNN-based methods (GCN, GAT, GraphSAGE) is substantially smaller than the overlap among GNN-based methods themselves. The oracle upper bound of GAC and GNN indicates complementary predictive performance. To enhance interaction performance, we introduce a Structural Context Tool (SCT) that augments the original tabular graph data with GNN predictions and topic summaries by aggregating over K-hop neighbors or random walks. The SCT can generate augmentation during graph inference. Across ten datasets, our method achieved an average improvement of 3.91% in node classification accuracy compared to baseline GAC. Further analytical experiments indicate that LLM-Graph interactions in tabular form require both the global data bias from GNN training and the local structural summary from neighbor aggregation. Executor code and preprocessed dataset are available at the `https://anonymous.4open.science/r/anonymous-artifact-C7A2`.

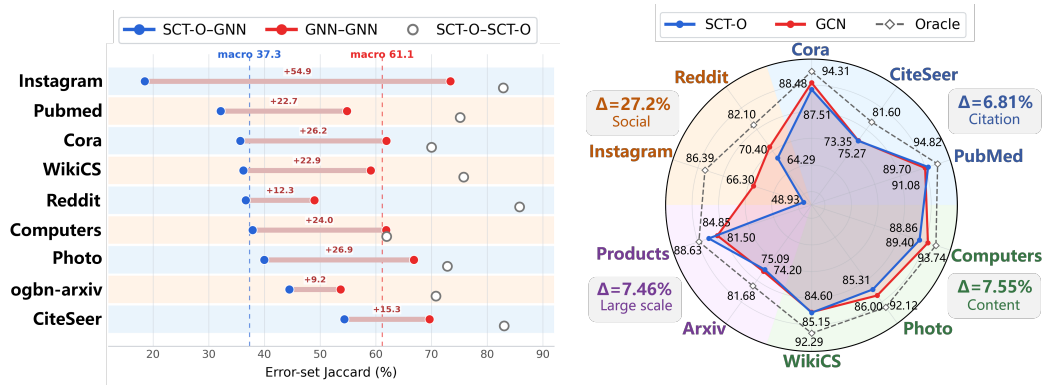
1 INTRODUCTION

Large language models serve as the foundation for intelligent agents. They possess capabilities such as long-context understanding, chain-of-thought reasoning, and tool use, along with strong generalization and transfer abilities, and are widely applied in fields such as natural language processing, video, image processing, and code generation. Recently, numerous studies have applied large language models to the field of graph reasoning, including tasks such as node classification, link prediction, and knowledge graph reasoning. Node classification serves as a foundational component in downstream applications such as social networks and recommendation systems.

Graph neural networks constitute a dominant paradigm for graph learning, which encode structural information and global dataset biases through message passing mechanisms; however, adjusting the numerous parameters involves significant time costs. For text-attributed graphs (Ye et al., 2024) such as e-commerce, web links, and social networks, LLMs can leverage the generalization and reasoning capabilities acquired through pre-training to directly perform graph reasoning.

Unlike sequential data, graph data cannot be directly adapted to the input format of large language models and requires interaction through transformation interfaces such as representation alignment, prompt enhancement, tool use, or code generation. Graph-as-Code (Finkelshtein et al., 2026) systematically investigates the interaction patterns between LLMs and graphs, demonstrating that table-based code generation is the most effective approach for eliciting an LLM's graph reasoning capabilities compared to direct methods, thereby guiding practical applications in specific downstream task scenarios. Although GAC can access neighbor information, this does not guarantee the formation of effective structural representations, nor does it provide a global bias that references the dataset

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) error-set overlap between pairs of methods

(b) accuracy and the oracle upper bound

Figure 1: Complementarity between SCT-O and GNN-based models. SCT-O refers to the GAC results we reproduced. (a) Error-set Jaccard across method pairs. Error-set Jaccard means $J(A, B) = |E_A \cap E_B|/|E_A \cup E_B|$. Each method consists of five experiments with independent random seeds. (b) Test accuracy of SCT-O, GCN, and their oracle upper bound of two methods. The oracle accuracy is the percentage of targets correctly classified by at least one method.

as a whole. In contrast, trained graph models explicitly encode these properties through message passing. In addition, GNNs now have stronger baseline options. Tuned-GCN (Luo et al., 2024) shows that carefully tuned classical GNNs remain strong baselines against other architectures such as Graph Transformer.

Two previous studies have found that GNNs and LLMs can complement each other. GNN-as-Judge (Xu & Ding, 2026) uses predictions from both LLMs and GNNs as pseudo-labels for weakly supervised fine-tuning, while GLANCE (Loveland et al., 2026) uses lightweight routing to decide when an LLM should assist a GNN. As shown in Fig. 1(a), we reproduced the GNN baseline results using the fine-tuned parameters from Tuned-GCN and reproduced the SCT-O results using the original prompt provided by GAC. We analyzed the error-set coverage of both approaches and found that, regardless of the dataset, the error-set overlap within the GNN methods (GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017)) was significantly higher than the overlap between any GNN method and the GAC method. In Fig. 1(b), we further analyzed the Oracle upper bounds for the node classification accuracy of GAC and GNN. This indicates that the GNN and the GAC are highly complementary for graph inference. We suggest that the upper bounds for fusion primarily stem from two aspects: the local structure summary by aggregating neighbors of target node and the global dataset bias during GNN training.

To investigate which factors determine the effectiveness of graph augmentation in code-based LLM inference, we introduce Structural Context Tools (SCT) and study four dimensions: (1) *information type*: topic tuple augmentation versus GNN predictions (predicted class, confidence, margin, and entropy); (2) *augmentation strategy*: direct exposure of node-level augmentation versus structured aggregation over K-hop neighbors or random-walk neighbors; (3) *model configuration*: model scale and inference intensity; (4) *feature characteristics*: textual and structural properties, such as text length, semantic similarity, node degree, and graph homophily.

Our evaluation yields three main insights:

- Based on the complementary between GAC and GNNs in node classification, we propose a Structural Context Tool that leverages GNN predictions and topic tuple, combined with a neighbor aggregation strategy, to enhance the graph inference capabilities of LLM agents.
- SCT achieved an average improvement of 3.91% across ten datasets. Our analysis revealed that the primary benefits of SCT stem from the local structural summary from neighbor aggregation and the global data bias during GNN training.
- Further analysis indicates that GNN predictions are the most stable enhancement component, while neighbor aggregation is also effective. And the effectiveness of Topics Tuple relies on the quality of the text in the dataset.

2 Related Work

2.1 Graph Methods

Graph Neural Networks. Classical GNNs GCN, GAT and GraphSAGE established message passing as a standard paradigm for graph representation learning. Subsequent work has focused on redefining neighbors and enriching message propagation. Geom-GCN (Pei et al., 2020), SPNs (Abboud et al., 2023), and Co-GNN (Finkelshtein et al., 2024) exploit latent geometry, shortest-path information, and adaptive communication strategies, respectively, while NBFNet (Zhu et al., 2021) performs path-based message aggregation for both homogeneous and heterogeneous graphs. Beyond model design, Pitfalls (Shchur et al., 2019) highlight the sensitivity of GNN evaluation to dataset splits and benchmark protocols.

Graph Foundation Models. Aiming to improve cross-graph generalization through pretraining, TS-GNNs (Finkelshtein et al., 2025) exploit permutation symmetries over nodes, labels, and features to generalize attribute prediction to unseen graphs. Flock (Kim et al., 2026) constructs probabilistic path representations based on node and relation equivariance for knowledge graph link prediction, while ULTRA (Galkin et al., 2024) builds relation-level adjacency graphs to enable zero-shot transfer across knowledge graphs.

2.2 LLM–Graph Interaction

Hybrid LLM–GNN Methods. Hybrid approaches combine the semantic reasoning capabilities of LLMs with the structural inductive bias of GNNs, typically through an alignment interface. GraphToken (Perozzi et al., 2024) compresses graph structure into continuous prompt representations, while GraphGPT (Tang et al., 2024) aligns graph and language embeddings through instruction tuning. LLaGA (Chen et al., 2024) serializes local neighbors into structure-preserving sequences, and LangTopo (Guan et al., 2024) aligns topological information with natural-language tokens. UniGraph (He et al., 2025) uses textual node attributes to support unified cross-domain pretraining through masked graph learning. EstGraph (Maurya & Liu, 2026) further incorporates statistics derived from random walks into prompts to support reasoning over very large graphs.

Graph Prompting. To convert graph data into sequence data, GraphTalk (Fatemi et al., 2024) studies how encoding schemes, tasks, and graph structures affect graph prompting. InstructGLM (Ye et al., 2024) learns generative graph models through multi-task and multi-prompt instruction tuning, and SOG (Wu et al., 2026) aligns discrete structural and semantic tokens through structured question answering. Structural Prompt (Huang et al., 2024a) suggests that LLMs may rely more on label-related semantic cues than on explicitly encoded topology; (Guan et al., 2025) show that attention between node tokens does not necessarily reflect graph connectivity; and (Dai et al., 2025) find that graph reasoning performance is highly sensitive to input format and solution strategy. GraphText (Zhao et al., 2024) maps graph structure and attributes into a unified textual space. GraphOmni (Xu et al., 2026) systematically compares graph types, serialization formats, and prompt designs. TAPE (He et al., 2024) uses LLM-generated predictions and explanations to enrich node features for downstream GNN reasoning.

Graph Tools. Tool-based approaches allow LLMs to invoke external graph models or graph-processing operations during reasoning. Graph-ToolFormer (Zhang, 2023) exposes graph loading, attribute computation, and task-specific graph operations as callable tools, and GRASP (Guo et al., 2026) alternates on-demand neighbor retrieval with code execution to explore graph structure and perform deterministic computation.

Code Generation for graph. A closely related direction represents graph reasoning as executable programs rather than static textual descriptions. CodeGraph (Cai et al., 2024) uses a small number of program demonstrations to guide LLMs in solving graph tasks through code generation, while LLM4Graph (Li et al., 2024) invokes existing graph-processing libraries to augment graph analysis. Graph-as-Code (Finkelshtein et al., 2026) introduces an iterative generate–execute–observe paradigm, in which the LLM writes programs over tabular graph data and accesses node text, graph structure, and visible labels on demand.

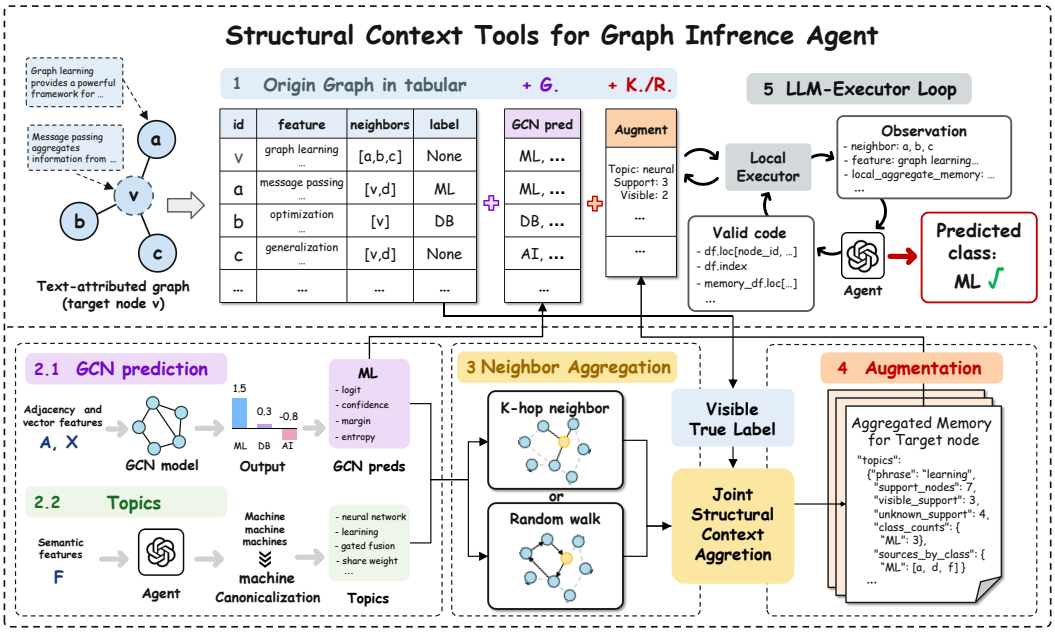


Figure 2: Overall flowchart of the LLM agent and Structural Context Tool interaction.

3 PRELIMINARIES

Node Classification. Let $\mathcal{T}$ be the set of all finite token sequences. $\mathcal{G} = (V, E)$ denote a undirected graph with $N = |V|$ nodes, adjacency matrix $A \in \{0,1\}^{N \times N}$ with finite text feature $x_u \in \mathcal{T}$ and class $\mathcal{Y}_u \in \{1, \ldots, C\} \subset \mathcal{T}$ of node u. We consider a fixed node classification task, where the graph structure and node texts are accessible during graph inference. Visible labels are only available for the training subset $V_{\text{vis}} \subset V$; labels of validation and test nodes are not exposed during inference.

Tabular Graph Data. We follow the GAC setting that the LLM accesses graph data through a tabular interface containing three fields: `features`, `neighbors`, and `label`. The `features` field stores node text x_u, `neighbors` provides adjacent node identifiers, and `label` contains y_u for $u \in V_{\text{vis}}$ or `None` otherwise. Further augmentation statistics and label-based aggregation are computed using visible labels only. Validation labels are used exclusively for selecting GCN hyperparameters, while test labels are reserved for final evaluation.

Graph Inference. LLM-Graph interaction models : $\mathcal{T} \times \mathcal{T}^N \times A \to \mathcal{T}$. Each mode encodes the interaction history in $\mathcal{T}$, node text feature in $\mathcal{T}^N$ and adjacency matrix. the output can provide predicted label $\hat{Y} \in \{1, \ldots, C\}$. For each target node, the LLM independently interacts with the tabular graph data through pandas code, receives observations generated by the executor, and finally outputs the class label. Query histories are isolated across targets, and the executor enforces the information boundary. All methods use the same dataset splits and node identifiers to ensure a controlled comparison.

4 METHODS

4.1 SCT INTERACTION FLOW

As shown in Figure.2, SCT varies the augmentation available through a shared read-only query interface. SCT-O exposes node text, neighbors and visible training labels. SCT-G additionally exposes frozen, dataset-specific GCN outputs (`logit`, `confidence`, `margin`, and `entropy`) for each node as pseudo-labels. The GCN models follow the strong baseline Tuned-GCN (Luo et al., 2024), whose outputs are predictive augmentation, not observed labels. SCT-K and SCT-R retain the SCT-G fields and further provide target-specific aggregated augmentation. They use the same extraction and aggregation procedure, differing only in how neighbor nodes are selected.

4.2 SHARED AUGMENTATION AGGREGATION

For target v, let S_v be its selected set of distinct neighbor nodes, excluding v. Each neighbor u has a normalized topic set T_u extracted from its text without labels or the class inventory. $\mathbb{I}[\cdot]$ is an indicator function that indicates whether an element belongs to a specific set. For topic t and class y, visible label augmentation is

$$c_v(t) = \sum_{u \in S_v} \mathbb{I}[t \in T_u], \quad b_v(y) = \sum_{u \in S_v \cap V_{\text{vis}}} \mathbb{I}[y_u = y], \quad c_v(t, y) = \sum_{u \in S_v \cap V_{\text{vis}}} \mathbb{I}[t \in T_u]\mathbb{I}[y_u = y]. \quad (1)$$

Thus, topic support $c_v(t)$ is computed from all selected neighbors, whereas visible label associations and marginals use only visible training labels. The joint count $c_v(t, y)$ is essential because topic and label marginals alone do not reveal which visible classes support a particular semantic topic. A neighbor without a visible training label still contributes to topic support $c_v(t)$, but cannot contribute to joint count $c_v(t, y)$ or visual label support $b_v(y)$.

For GCN prediction augmentation, partition S_v into $S_v^{\text{vis}} = S_v \cap V_{\text{vis}}$ and $S_v^{\text{unk}} = S_v \setminus V_{\text{vis}}$. For any group $g \in \{S_v^{\text{vis}}, S_v^{\text{unk}}\}$, we compute

$$a_v(y; g) = \sum_{u \in S_v^g} \mathbb{I}[\hat{y}_u = y], \qquad c_v(t, y; g) = \sum_{u \in S_v^g} \mathbb{I}[t \in T_u]\mathbb{I}[\hat{y}_u = y], \quad (2)$$

where $\hat{y}_u$ is the predicted class by GCN. Mean GCN confidence is recorded by group and predicted class. Visible labels and GCN predictions statistics remain separate; no unknown labels are inferred. The LLM queries these summaries and makes the final decision without a prescribed fusion rule.

4.3 K-HOP NEIGHBOR AGGREGATION

SCT-K uses the complete neighbor within a specified graph distance k. For target node v, its neighbor set is

$$S_v^K(k) = \{u \in V \setminus \{v\} : d(u, v) \leq k\}, \quad (3)$$

where $d(u, v)$ is the shortest-path distance. The main experiments set $k = 1, 2$, so the neighbor set contains every distinct neighbor of v. The target itself is excluded, and no sampling or neighbor-count cap is applied.

4.4 RANDOM WALK NEIGHBOR AGGREGATION

SCT-R selects neighbors using a random walk with restarts. The walk begins at v and, at each step, either returns to v with probability r or moves uniformly to a neighbor of the current node. After B steps, let $\text{Freq}_{RW}(u)$ be the number of visits to node u. We select

$$S_v^R = \text{Top}_{K_s}\left(\{u \in V \setminus \{v\} : \text{Freq}_{RW}(u) > 0\}\right), \quad (4)$$

where Top_{K_s} returns up to K_s distinct nodes ranked by decreasing visit frequency. The main experiments use $B = 300$, $r = 0.15$, and $K_s = 12$. Ranking ties and walk seeds are specified in Appendix B.

Visit frequency determines which nodes enter the neighbor set; it does not weight their topic, visible label, or GCN prediction statistics. Unlike SCT-K, this rule imposes a neighbor budget while allowing selected nodes to extend beyond one hop. Both methods pass their resulting neighbor sets to the same augmentation aggregator. Their comparison therefore evaluates the full neighbor aggregation rules, including differences in reach and neighbor size.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Models and metrics. The classification model is DeepSeek-V4-Flash with high-effort thinking, temperature zero, a maximum output of 8,192 tokens, and at most eight augmentation-query steps

Table 1: Node-classification accuracy (%). Values are reported as mean ± standard deviation. The top **1st**, **2nd** and **3rd** results are highlighted. GAC* is result reported in original paper. Or.† refers to the oracle upper bound of GCN and SCT-O, which is correct whenever either GCN or SCT-O predicts the target node correctly.

Method	Cora	CiteSeer	PubMed	Comp.	Photo	WikiCS	arxiv	products	Instagram	Reddit
GCN	88.48 ± 0.67	73.35 ± 0.65	89.70 ± 0.35	89.40 ± 0.45	86.00 ± 0.37	84.60 ± 0.62	74.20 ± 0.12	81.50 ± 0.21	**66.30** ± 3.05	**70.40** ± 1.47
GAT	88.25 ± 0.93	76.59 ± 1.02	89.23 ± 0.21	89.30 ± 0.67	86.90 ± 0.44	85.73 ± 0.74	73.30 ± 0.18	80.99 ± 0.16	**66.53** ± 2.25	**68.77** ± 0.64
SAGE	88.44 ± 0.56	75.13 ± 1.37	90.30 ± 0.17	89.37 ± 0.15	86.73 ± 0.21	85.93 ± 0.55	73.00 ± 0.28	83.89 ± 0.36	**68.73** ± 0.67	67.87 ± 0.40
GAC*	85.16 ± 1.47	71.80 ± 2.22	89.90 ± 1.85	86.20 ± 3.55	86.40 ± 2.65	82.20 ± 3.63	74.40 ± 3.02	82.70 ± 2.66	56.40 ± 2.56	61.60 ± 2.36
SCT-O	87.51 ± 0.58	75.27 ± 0.34	91.08 ± 0.37	88.86 ± 1.27	85.31 ± 0.66	85.15 ± 1.25	75.09 ± 1.03	84.85 ± 1.14	48.93 ± 1.17	64.29 ± 2.00
SCT-G	**90.58** ± 0.37	**76.69** ± 0.84	**91.66** ± 0.73	**90.32** ± 1.09	**88.17** ± 0.67	**87.40** ± 0.94	**75.58** ± 1.83	**86.13** ± 0.76	58.36 ± 1.15	**68.01** ± 2.25
SCT-K	**90.56** ± 0.33	**77.44** ± 0.26	**91.30** ± 0.70	**90.40** ± 0.40	**87.70** ± 0.70	**88.02** ± 0.94	**75.10** ± 0.91	**85.27** ± 0.61	59.00 ± 1.70	66.63 ± 2.77
SCT-R	**91.80** ± 0.28	**78.08** ± 0.26	**92.20** ± 0.72	**91.10** ± 0.51	**88.00** ± 0.36	**87.53** ± 1.07	**76.70** ± 2.99	**85.40** ± 0.17	61.20 ± 0.20	63.20 ± 3.90
Or.†	94.31 ± 0.08	81.60 ± 0.48	94.82 ± 0.41	93.74 ± 0.88	92.12 ± 0.41	92.29 ± 0.67	81.68 ± 1.50	88.63 ± 0.36	86.39 ± 0.74	82.10 ± 0.92

per target. Topic extraction uses Qwen3.8-Flash with temperature zero and a 4,096 token output limit. Full model and runtime settings are given in Appendix B.

Accuracy is the percentage of fixed test targets classified correctly, with unresolved episodes counted as incorrect. We report its mean and standard deviation over five runs. For two methods A and B evaluated on matched targets, let E_A and E_B denote their error sets. Error-set Jaccard is $J(A, B) = |E_A \cap E_B|/|E_A \cup E_B|$. Their pairwise Oracle accuracy is the percentage of targets correctly classified by at least one method.

Datasets and protocol. We evaluate node classification on ten text-attributed graphs: the citation networks Cora and CiteSeer (Sen et al., 2008), and PubMed (Namata et al., 2012); WikiCS (Mernyei & Cangea, 2020); ogbn-arxiv and ogbn-products (Hu et al., 2020); and the Computers, Photo, Instagram, and Reddit text-attributed versions distributed through LLMNodeBed (Wu et al., 2025). Text fields, graph preprocessing, and the fixed splits used here are documented in Appendix A.3 and Table 6. Target sampling, model settings, and paired diagnostic protocols are detailed in Appendix A.1.

Experimental configuration. Table 5 summarizes the frozen experimental configuration used throughout the main study. Unless otherwise specified, the same configuration is shared across datasets. The fixed values are intended to provide a common evaluation protocol rather than dataset-specific optima. Topics are cached by neighbor node and reused across methods and evaluation runs. SCT-G, SCT-K, and SCT-R use the same frozen, precomputed GCN predictions; the GCN is not retrained across evaluation seeds.

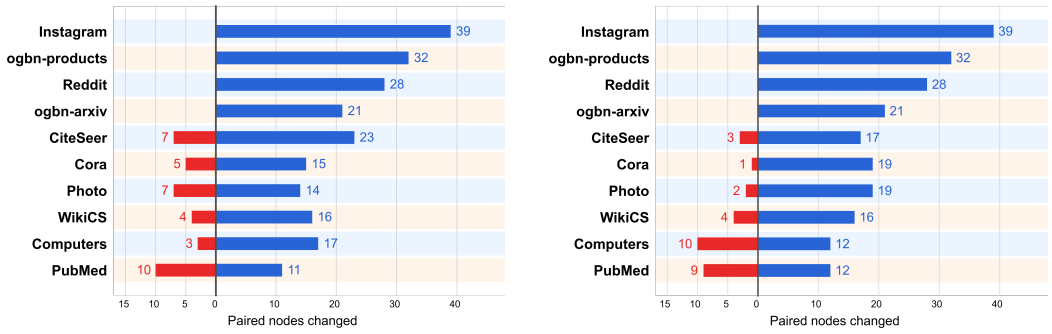
5.2 Main Results

We evaluate on ten text-attributed graphs using fixed data splits and matched test targets across SCT variants. Table 1 reports mean accuracy and standard deviation over five evaluation runs; all the variations maintain the same data splits and prepared augmentation.

Finding 1. *SCT can improve the performance of LLM-Graph interactions in tabular form, but it is still quite far from Oracle's upper bound.*

Table 1 compares the four SCT configurations with classical graph models and the Graph-as-Code (Finkelshtein et al., 2026) reference across all ten datasets. SCT-O retains the original code-based interface, while SCT-G additionally exposes GNN predictions for each node. SCT-K and SCT-R further introduce the neighbor augmentation aggregation framework, using the complete one-hop neighbors and random walk selected neighbors, respectively. Thus, SCT-K and SCT-R share the same augmentation representation and differ only in augmentation strategy.

All SCT configurations follow the same frozen data splits and target sampling protocol, enabling controlled comparisons across augmentation settings. Overall, at least one SCT configuration achieves the highest mean accuracy on eight of the ten datasets. In particular, SCT-G consistently improves over SCT-O on all ten datasets, showing that graph-derived predictive augmentation pro-



(a) SCT-K (b) SCT-R

Figure 3: Paired effect of GNN predictions on the evaluation set. Each full method is compared with its matched variant without GNN predictions. Blue bars count repaired predictions; red bars count harmed predictions. Each comparison uses the same evaluation nodes and neighbor aggregation.

Table 2: Component ablation on four representative datasets. Full combines GNN predictions, neighbor aggregation, and topic augmentation. Each "w/o" variant removes the indicated component while retaining the other two; w/o All corresponds to SCT-O.

	Cora		Computers		ogbn-products		Reddit	
Variant	K	R	K	R	K	R	K	R
Full	90.56	91.80	90.40	91.10	85.27	85.40	68.63	63.20
w/o Aggregation	87.54	87.54	88.66	88.66	82.70	82.70	62.13	62.13
w/o Graph	88.70	88.45	89.70	90.18	81.31	82.37	64.67	60.68
w/o Topic	90.53	88.89	89.72	90.01	85.57	85.68	61.30	56.46
w/o All	87.51	87.51	88.86	88.86	84.85	84.85	64.29	64.29

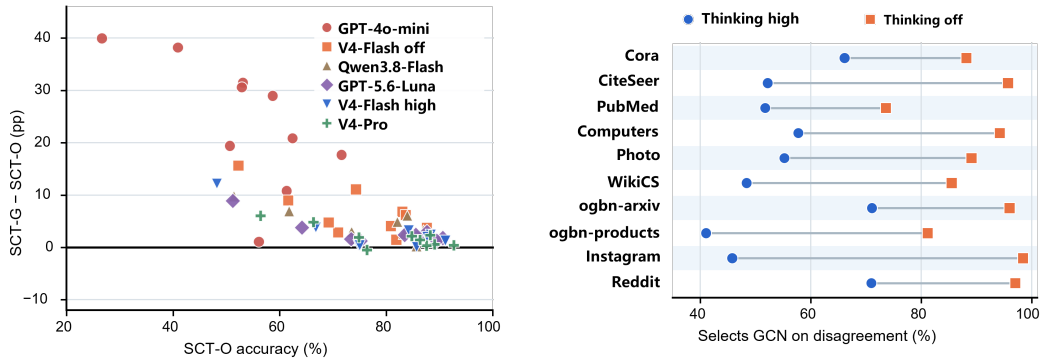
vides a broadly useful complement to the original tabular graph data. The aggregation variants further improve performance on several datasets, but their relative effects are substantially more heterogeneous: changing the neighbor set can yield additional gains, little change, or even degradation depending on the underlying graph. This contrast is especially visible on the two social-network datasets. Although the SCT variants bring gains to the LLM with the same trend as on the other datasets, SCT does not manage to outperform classical GNNs on Instagram and Reddit.

5.3 Contribution of Augmentation Components

Finding 2. *The benefits of SCT stem primarily from the global data biases captured during GNN training and the local structures Summary resulting from neighbor aggregation. GNN predictions and neighbor aggregation provide broad gains, while the effects of topic tuple depend on dataset.*

We conduct matched component ablations on four representative datasets. Full combines GNN predictions, structured aggregation, and topic-conditioned augmentation. w/o Graph removes all GNN prediction fields and statistics; w/o Topic removes topic-conditioned augmentation; w/o Aggregation exposes only the underlying topic tuples and GNN predictions queries without constructing aggregated statistics. w/o All reduces to SCT-O.

Table 2 shows that removing GNN predictions consistently reduces accuracy, with the largest drop on ogbn-products. Removing aggregation also degrades performance across all four datasets, indicating that direct access to Topic and Graph queries does not replace structural augmentation. Topic augmentation is less uniform: it is strongly beneficial on Reddit and generally helpful on Cora and Computers, while slightly reducing accuracy on ogbn-products. We further analyze GNN predictions at the node level using the fixed evaluation set. Figure 3 decomposes the paired changes into *repairs* and *harms* on the set. Repairs substantially outnumber harms for both SCT-K and SCT-R, providing a decision view of the five random seed gains reported above.



(a) (b)

Figure 4: (a) SCT-G and SCT-O accuracy differences across six LLMs and ten datasets; each point represents one LLM configuration evaluated on one dataset. (b) Fraction of cases in which SCT-G selects the GCN pseudo-label under DeepSeek-V4-Flash thinking high and thinking off.

Table 3: Diagnostic results on four WebKB datasets. Values are node-classification accuracy (%).

Dataset	Raw GCN	SCT-O	SCT-G	SCT-K	SCT-R
Texas	55.26	**100.00**	**100.00**	**100.00**	97.37
Cornell	53.85	**97.44**	89.74	87.18	96.58
Washington	54.35	93.48	**95.65**	93.48	93.48
Wisconsin	49.06	**94.34**	86.79	88.68	93.48

5.4 Diagnostic Results on Heterophilic Web Graphs

Finding 3. *LLMs perform well on heterophilic datasets with long, high-quality text, whereas GCNs have limited performance on these datasets. SCT is not applicable in this scenario.*

We additionally evaluate SCT on four WebKB graphs as heterophilic graphs. Dataset statistics are given in Appendix A.3.These datasets exhibit a substantially different regime from the main benchmarks: the original SCT-O interface already achieves 93–100% accuracy, whereas the standalone GCN obtains a macro-average accuracy of only 53.13%.

Under this setting, additional GNN prediction augmentation does not provide consistent gains. The macro-average accuracies of SCT-O, SCT-G, SCT-K, and SCT-R are 96.31%, 93.05%, 92.33%, and 95.66%, respectively. This result suggests that when the text is semantically rich enough, GNN-based prediction enhancement is ineffective.

5.5 SCT Effects with Different LLMs

The main experiments use DeepSeek-V4-Flash with high thinking. We examine additional LLMs with SCT to evaluate how SCT performance varies with model capability.

Finding 4. *GNN predictions produce different accuracy gains across LLM configurations, with the largest observed gains occurring when SCT-O accuracy (model capability) is relatively low.*

Figure 4(a) shows this variation across classifier and dataset pairs. On ogbn-arxiv, for example, GPT-4o-mini improves by 39.97% from an SCT-O accuracy of 26.63%, whereas DeepSeek-V4-Pro changes by 0.47 % from 76.37%.

Finding 5. *When SCT-O disagrees with GCN, SCT-G selects the GCN class more frequently with thinking off than with thinking high.*

For each setting, we consider targets on which SCT-O and GCN predict different classes. Figure 4(b) shows a higher GCN selection rate with thinking off on all ten datasets. Across these conditional sets, SCT-G selects the GCN class in 92.11% of 2,066 cases with thinking off, compared with

Table 4: Change in topic support from SCT-K to SCT-R on the target set ($\Delta = R - K$). Neighbor counts distinct selected neighbor nodes. Train$\geq k$ is the percentage of targets with at least k visible training label neighbor nodes. Visible≥ 2 and Pseudo≥ 2 are target macro percentages of retained topics with repeated same-class support from visible labels and GCN predictions, respectively. Ratio is Pseudo≥ 2 divided by Visible≥ 2 within each method.

Dataset(%)	Δ Neighbor	Δ Train≥ 1	Δ Train≥ 2	Δ Visible≥ 2	Δ Pseudo≥ 2	Δ Ratio
Cora	+7.39	+9.23	+37.64	+1.15	+2.32	-0.18
CiteSeer	+5.71	+15.83	+41.69	+0.42	+1.40	+0.11
PubMed	+7.89	+23.40	+63.10	+1.80	+4.14	-0.07
Computers	-1.86	+8.40	+26.40	-0.63	-1.38	+0.10
Photo	-6.84	+9.00	+25.50	-1.27	-2.47	+1.01
WikiCS	-26.82	+5.90	+14.40	-1.97	-4.52	+1.31
ogbn-arxiv	-0.81	+6.70	+18.60	-1.22	-2.66	-0.02
ogbn-products	-26.37	+11.70	+14.00	-5.75	-25.83	-0.47
Instagram	-2.20	+5.50	+17.50	-1.07	-2.23	+0.45
Reddit	+3.92	+5.30	+19.20	+0.02	+0.04	+0.20

53.71% of 1,696 cases with high-effort thinking. The pattern is consistent with greater reliance on the supplied GNN prediction when thinking is disabled. Furthermore, the more reasoning capable the model is, the better it can match the prediction results of GAC and GNN using SCT.

5.6 Augmentation Support under neighbor aggregation

Finding 6. *Compared to k-hop, the random walk can handle extremely large graphs, identify more nodes with visible labels, and improve topic support ratio.*

We compare the topic support of SCT-K complete one-hop neighborhood and SCT-R random walk on the same targets. For a retained topic t, Visible≥ 2 and Pseudo≥ 2 report the target percentages of retained topics meeting these criteria. Table 4 reports changes from SCT-K to SCT-R,

$$\max_y c_v(t, y) \geq 2, \qquad \max_y \textstyle\sum_g c_v^G(t, y; g) \geq 2.$$

SCT-R increases the fraction of targets with at least one neighbor containing visible label on all ten datasets. The increase in Train≥ 2 is especially large on sparse citation graphs: $+37.64$ percentage points on Cora, $+41.69$ on CiteSeer, and $+63.10$ on PubMed. Additional training anchors do not always form repeated support for the same topic and class. Visible≥ 2 and Pseudo≥ 2 both decrease on six datasets. For example, on ogbn-products, SCT-R uses 26.37 fewer sources per target on average and Pseudo≥ 2 falls by 25.83 points. These comparisons describe the topic support; they do not isolate source quality from differences in source-set size or retained topics. Furthermore, pre-training or fine-tuning a model can also cause an LLM to acquire global data biases.

6 Conclusion

Large models can interact with text-based property graph data in the form of code and tables through queries. We analyzed the error-set overlap between the Graph-as-Code method and the GNN method on the node classification task, computed the oracle upper bounds for both, and clarified the complementarity of their predictions. Based on these findings, we propose a structural context tool that can be invoked by LLMs. This tool fuses GNN predictions with textual themes, utilizes neighborhood aggregation to generate effective reasoning evidence, and interacts with large language models through a dedicated interface. We achieved a broad improvement of 3.91% across ten common graph datasets and analyzed the effectiveness of each module. GNN predictions provide effective pseudo-labels, while neighborhood aggregation makes it easier for the model to distinguish the applicable scopes of predictions from both GNN and GAC. LLM-Graph interactions in tabular form still have significant room for improvement and represent a promising direction for future research.

AI USE STATEMENT

Generative AI tools serve as the experimental components in this paper and are used for language polishing. All technical arguments, experimental results, code, and references are independently verified by the authors.

REFERENCES

Ralph Abboud, Radoslav Dimitrov, and İsmail İlkan Ceylan. Shortest path networks for graph property prediction. In *LoG*, 2023.

Qiaolong Cai, Zhaowei Wang, Shizhe Diao, James Kwok, and Yangqiu Song. Codegraph: Enhancing graph reasoning of llms with code. *arXiv*, 2024.

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

Mark Craven, Dan DiPasquo, Dayne Freitag, Andrew McCallum, Tom Mitchell, Kamal Nigam, and Se
'an Slattery. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 118(1–2):69–113, 2000.

Xinnan Dai, Haohao Qu, Yifei Shen, Bohang Zhang, Qihao Wen, Wenqi Fan, Dongsheng Li, Jiliang Tang, and Caihua Shan. How do large language models understand graph patterns? a benchmark for graph pattern comprehension. In *ICLR*, 2025.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *ICLR*, 2024.

Ben Finkelshtein, Xingyue Huang, Michael Bronstein, and İsmail İlkan Ceylan. Cooperative graph neural networks. In *ICML*, 2024.

Ben Finkelshtein, İsmail İlkan Ceylan, Michael Bronstein, and Ron Levie. Equivariance everywhere all at once: A recipe for graph foundation models. *arXiv*, 2025.

Ben Finkelshtein, Silviu Cucerzan, Sujay Kumar Jauhar, and Ryen White. Actions speak louder than prompts: A large-scale study of llms for graph inference. In *International Conference on Learning Representations*, volume 2026, pp. 124818–124840, 2026.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jVEoydFOl9.

Zhong Guan, Hongke Zhao, Likang Wu, Ming He, and Jianpin Fan. Langtopo: Aligning language descriptions of graphs with tokenized topological modeling, 2024.

Zhong Guan, Likang Wu, Hongke Zhao, Ming He, and Jianpin Fan. Attention mechanisms perspective: Exploring llm processing of graph-structured data. In *ICML*, 2025.

Xiaojun Guo, Mingxue Tian, Chenheng Zhang, Xiaohan Wang, Jiajun Chai, Guojun Yin, Wei Lin, Yifei Wang, and Yisen Wang. Grasp: Graph reasoning via agentic solving and probing of llms. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=nWOjSTq1ae.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

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. In *ICLR*, 2024.

Yufei He, Yuan Sui, Xiaoxin He, and Bryan Hooi. Unigraph: Learning a unified cross-domain foundation model for text-attributed graphs. In *KDD*, 2025.

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

Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. Can llms effectively leverage graph structural information through prompts, and why? In *TMLR*, 2024a.

Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. Can gnn be good adapter for llms? In *Proceedings of the ACM Web Conference*, 2024b. URL https://arxiv.org/abs/2402.12984.

Jinwoo Kim, Xingyue Huang, Krzysztof Olejniczak, Kyungbin Min, Michael Bronstein, Seunghoon Hong, and Ismail I Ceylan. Flock: A knowledge graph foundation model via learning on random walks. In *International Conference on Learning Representations*, pp. 36664–36705, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/3ddbd473456a57e3cafb1ee51ddf8ff6-Paper-Conference.pdf.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Xin Li, Weize Chen, Qizhi Chu, Haopeng Li, Zhaojun Sun, Ran Li, Chen Qian, Yiwei Wei, Chuan Shi, Zhiyuan Liu, et al. Can large language models analyze graphs like professionals? a benchmark, datasets and models. In *NeurIPS*, 2024.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.00149.

Donald Loveland, Yao-An Yang, and Danai Koutra. Glance for context: Learning when to leverage llms for node-aware gnn-llm fusion. In *International Conference on Learning Representations*, volume 2026, pp. 90402–90428, 2026.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic gnns are strong baselines: Reassessing gnns for node classification. In *Advances in Neural Information Processing Systems*, volume 37, pp. 97650–97669, 2024. doi: 10.52202/079017-3098. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/b10ed15ff1aa864f1be3a75f1ffc021b-Paper-Datasets_and_Benchmarks_Track.pdf.

Sunil Kumar Maurya and Xin Liu. Evaluating LLMs on large-scale graph property estimation via random walks. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 39749–39777, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1846. URL https://aclanthology.org/2026.acl-long.1846/.

Peter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv*, 2020.

Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *Proceedings of the Workshop on Mining and Learning with Graphs*, 2012. URL https://berthuang.com/papers/namata-mlg12.pdf.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of EMNLP-IJCNLP*, pp. 188–197, 2019. URL https://aclanthology.org/D19-1018/.

Hongbin Pei, Bingzhe Wei, Kevin C.-C. Chang, Yizhou Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2020.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. Let your graph do the talking: Encoding structured data for llms. *arXiv*, 2024.

Prithviraj Sen, Galen Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. In *AI Magazine*, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation, 2019.

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. *SIGIR*, 2024.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Jingyao Wu, Bin Lu, Zijun Di, Xiaoying Gan, Meng Jin, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. $\langle \mathrm{SOG}_k \rangle$: One LLM token for explicit graph structural understanding, 2026. URL https://arxiv.org/abs/2602.01771.

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. In *ICML*, 2025.

Hao Xu, Xiangru Jian, Xinjian Zhao, Wei Pang, Chao Zhang, Suyuchen Wang, Qixin Zhang, Zhengyuan Dong, Joao Monteiro, Bang Liu, Qiuzhuang Sun, and Tianshu Yu. GraphOmni: A comprehensive and extensible benchmark framework for large language models on graph-theoretic tasks, 2026. URL https://arxiv.org/abs/2504.12764.

Ruiyao Xu and Kaize Ding. Gnn-as-judge: Unleashing the power of llms for graph learning with gnn feedback, 2026. URL https://arxiv.org/abs/2604.08553.

Hao Yan, Chaozhuo Li, Rui Long, Chao Yan, Jie Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Wei Han, Hao Sun, Wei Deng, Qi Zhang, Lichao Sun, Xing Xie, and Senzhang Wang. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/37d00f567a18b478065f1a91b95622a0-Abstract.html.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Language is all a graph needs. In *ACL*, pp. 1955–1973, 2024. URL https://aclanthology.org/2024.findings-eacl.132/.

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

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael M Bronstein, Zhaocheng Zhu, and Jian Tang. Graphtext: Graph learning in text space. *arXiv*, 2024.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural Bellman-Ford networks: A general graph neural network framework for link prediction. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f6a673f09493afcd8b129a0bcf1cd5bc-Abstract.html.

A Experimental and Dataset Details

A.1 Evaluation Protocol

Datasets and target sampling. We evaluate on the ten text-attributed graphs summarized in Table 6. The train/validation/test splits and node identifiers are fixed throughout the study. For each evaluation seed in $\{40, 41, 42, 43, 44\}$, we sample at most 1,000 nodes from the fixed test split. Cora and CiteSeer have fewer than 1,000 test nodes, so their complete test sets are used in every run. Within each run, SCT-O, SCT-G, SCT-K, and SCT-R are evaluated on identical target nodes. SCT-R selects neighbors using a seed that is fixed independently for each dataset–target pair. Evaluation seeds therefore change target sampling without changing the neighbor set of a given target.

Statistical reporting and paired diagnostics. Main-table accuracies are reported as the mean and standard deviation across the five evaluation runs. These values characterize variation under fixed splits and a fixed GCN, rather than variation across independent data splits or GCN training runs. For these analyses, predictions are paired only for targets successfully completed by both compared conditions. The paired diagnostics characterize changes in decisions and do not replace the five-run accuracy estimates.

A.2 Experimental Configuration

Table 5: Frozen experimental configuration. Fixed parameters are shared across datasets unless otherwise noted.

Parameter	Configuration
Classifier	DeepSeek-V4-Flash, thinking high, $T = 0$
Maximum output	8192 tokens
Topic extractor	Qwen3.8-Flash, thinking medium, $T = 0$, 4096-token output limit
Topic extraction	1–4 topics per neighbor; class labels are not exposed to the extractor
Topic normalization	Unicode NFKC, casefolding, whitespace normalization, and within-node deduplication; exact string matching only
Displayed topics K_p	20
SCT-K neighbor set	All distinct one-hop neighbors; target excluded; no neighbor cap
SCT-R walk budget B	300 steps
Restart probability r	0.15
SCT-R neighbor budget K_s	Top 12 distinct visited nodes; target excluded
SCT-R ranking	Visit count descending, with node ID as the deterministic tie-breaker
Random-walk seed	First eight bytes of SHA256(`dataset:target:1709`); independent of evaluation sampling seeds
Maximum augmentation steps	8
Observation limit	8000 characters
Decision fallback	Disabled
Execution	20 workers, 180-s timeout, at most three transport attempts
Evaluation sampling	Frozen data splits; target-sampling seeds $\{40, 41, 42, 43, 44\}$; at most 1000 test targets per dataset
GNN predictions	Dataset-specific GCN predictions are precomputed and frozen during LLM evaluation
Label visibility	Only training labels are visible as augmentation; validation and test labels are excluded from LLM augmentation construction

A.3 Dataset Statistics

Table 6 summarizes the ten evaluated graphs. Cora and CiteSeer have a history in collective classification (Sen et al., 2008); together with PubMed, they were used as citation-network benchmarks by Planetoid (Yang et al., 2016). The PubMed diabetes citation network is also described by Namata et al. (Namata et al., 2012). WikiCS originates from the Wikipedia graph benchmark (Mernyei & Cangea, 2020); its page text was subsequently collected for text-attributed graph evaluation (Liu et al., 2024). The Computers and Photo graphs draw on Amazon product and review data (Ni et al., 2019) and later processed text-attributed graph versions (Yan et al., 2023). The Instagram and Reddit user graphs were introduced in earlier LLM–graph work (Huang et al., 2024b). We use text-attributed versions aligned with LLMNodeBed (Wu et al., 2025), while ogbn-arxiv and ogbn-products are based on OGB (Hu et al., 2020). These references describe dataset provenance, not the particular splits or text preprocessing used here. Links in the table are unique undirected edges

Table 6: Statistics of the ten text-attributed graphs. Text coverage denotes the fraction of nodes with non-empty text.

Dataset	Nodes	Links	Cls.	Avg. deg.	Homo.	Avg. chars	Text cov.	Split
Cora	2,708	5,278	7	3.90	0.810	65.54	98.45%	60/20/20
CiteSeer	3,186	4,225	6	2.65	0.744	1,022.24	100%	60/20/20
PubMed	19,717	44,324	3	4.50	0.802	110.47	99.99%	60/20/20
Computers	87,229	628,274	10	14.41	0.823	492.77	100%	60/20/20
Photo	48,362	436,891	12	18.07	0.735	797.82	100%	60/20/20
WikiCS	11,701	215,603	10	36.85	0.654	3,215.56	100%	60/20/20
ogbn-arxiv	169,343	1,157,799	40	13.67	0.403	68.99	100%	official
ogbn-products	2,449,029	61,859,140	47	50.52	0.808	646.87	100%	official
Instagram	11,339	72,005	2	12.70	0.586	110.05	99.13%	60/20/20
Reddit	33,434	134,236	2	8.03	0.588	766.86	100%	60/20/20

Table 7: Statistics of the ten text-attributed graphs. Text coverage denotes the fraction of nodes with non-empty text.

Dataset	Nodes	Links	Cls.	Avg. deg.	Homo.	Avg. chars	Text cov.	Split
Texas	187	280	5	2.995	6.43	1,487.54	100%	59.9/19.8/20.3
Cornell	191	274	5	2.869	12.41	2,039.69	100%	59.7/19.9/20.4
Washington	229	365	5	3.188	15.07	1,597.53	100%	59.8/20.1/20.1
Wisconsin	265	459	5	3.464	18.08	2,109.54	100%	60.0/20.0/20.0

without self-loops. Homophily is computed offline from ground-truth labels solely for dataset characterization. For datasets without official splits, we freeze the train/validation/test partition before evaluation.

Text and label profiles. The node texts represent different kinds of objects in our processed inputs. Cora and PubMed use paper titles, CiteSeer contains titles and abstracts, and WikiCS uses preprocessed page content. Following the setup of GAC, our ogbn-arxiv and ogbn-product input contains titles only, despite the availability of more complete text. Computers and Photo use review text for the category classification of different product. Instagram uses short user biographies; Reddit uses concatenated user comments.

These differences affect how readily text and neighborhood summaries can support classification. A paper title may contain a discriminative method or disease term, but short text can omit the decisive detail. Longer reviews and encyclopedia text provide more context while also mentioning related objects and topics outside the target category. For Instagram and Reddit, the labels describe user role or popularity, which need not be stated by the topics users discuss. Thus, topic frequency should not be treated as a direct proxy for the class label.

Heterophilic WebKB datasets. The four graphs contain computer-science webpages from Texas, Cornell, Washington, and Wisconsin, originating from the WebKB collection (Craven et al., 2000). Texas, Cornell, and Wisconsin are also commonly used in heterophilic GNN evaluations (Pei et al., 2020); ACTIONS evaluates all four as text-attributed webpage graphs (Finkelshtein et al., 2026). Table 7 reports statistics computed from the GraphCom WebKB CSV files used in our O/G/K/R experiments, rather than statistics copied from those benchmarks. All nodes have page-level text in the `features` field. The pages mix titles, main content, navigation text, and sometimes material about multiple roles, courses, or projects. Each graph has the five semantic classes course, project, student, faculty, and staff, although integer class IDs are mapped separately for each dataset. Links are unique undirected edges after removing self-loops and duplicate or reciprocal records; homophily is the fraction of these links joining nodes with the same true class. The resulting edge counts and homophily need not match published versions, and our Texas text is shorter on average than that reported by ACTIONS.

A.4 Baseline Protocol Notes

The GCN baseline is trained separately for each dataset following the Tuned-GCN protocol. Validation labels are used only for GCN model selection and are never exposed to the LLM or to augmentation aggregation. GCN predictions are then precomputed and frozen for all nodes before SCT evaluation. Test labels are used exclusively for offline evaluation.

GAC is included as an external reference; its released settings on Computers and Instagram are not fully aligned with our frozen evaluation protocol and should therefore not be interpreted as strictly paired comparisons.

B augmentation Construction Details

This section provides implementation details of the augmentation construction procedure. Numerical hyperparameters are summarized in Table 5; here we clarify the preprocessing, counting, and neighbor-selection conventions used throughout the experiments.

B.1 Topic Extraction and Normalization

Topic extraction is performed once for each neighbor node with usable text and cached for subsequent target aggregation. The extractor receives only the node text and does not observe class labels or the class inventory. Each neighbor produces a small set of concise semantic topics; empty output is permitted when no usable topic can be extracted.

Extracted topics are normalized using Unicode NFKC normalization, case folding, and whitespace normalization, followed by within-node deduplication. Topic matching is exact after normalization; no embedding-based or fuzzy semantic matching is employed.

When the number of distinct topics exceeds the display budget, topics are ranked by decreasing total neighbor support, followed by decreasing visible support and lexicographic order. Topic truncation affects only the topic representation and does not remove neighbor nodes from neighbor statistics.

B.2 Aggregation and Counting Conventions

All aggregation statistics are computed over distinct neighbor nodes. A neighbor contributes at most once to each applicable count, regardless of repeated graph connections or, for SCT-R, repeated random-walk visits.

Neighbors without usable topics remain part of the selected neighbor set. They contribute to visible label and graph prediction marginals of neighbors but not to topic-conditioned statistics.

Visible label augmentation and GNN predictions are maintained as separate channels. Visible label statistics use only neighbors in V_{vis}, whereas statistics of graph predictions may include neighbors having visible or unknown labels. These two neighbor groups are disjoint. We do not construct a joint tensor of topics, visible labels, and GNN predictions, nor a confusion table of visible labels and predictions, during augmentation construction.

Graph prediction confidence is summarized independently of visible-label augmentation and is treated as model-derived predictive information rather than calibrated correctness probability.

B.3 augmentation strategy and Reproducibility

SCT-K uses all distinct one-hop neighbors of the target node, excluding the target itself, without imposing a neighbor-count limit. SCT-R instead samples graph context using the restarted random-walk procedure defined in Section 4.4. The walk configuration and neighbor budget are fixed across datasets and are reported in Table 5.

For SCT-R, repeated visits determine only which distinct nodes are selected. Once the neighbor set has been constructed, visit frequency does not weight topic, label, or graph prediction statistics. Ties in neighbor ranking are resolved deterministically by node identifier.

Random-walk randomness is seeded deterministically from the dataset and target identifiers and is independent of the target sampling seeds used for evaluation. Thus, each target receives a reproducible neighbor set under a fixed experimental configuration.

C K-HOP NEIGHBORHOOD SENSITIVITY

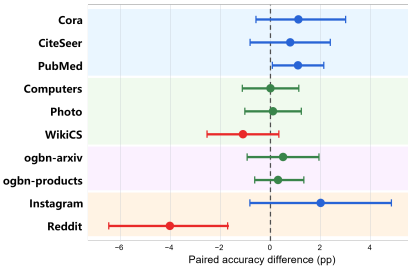
We compare complete one-hop and two-hop SCT-K neighborhoods on Cora, CiteSeer, PubMed, WikiCS, Instagram, and Photo. Both variants use the same topic, visible label, and GNN prediction aggregation; only the source radius changes. The other four datasets are excluded from this sensitivity study because complete two-hop neighbor aggregation exceeded the computational scope of these runs. We therefore do not extrapolate the results to all ten datasets.

Table 8: One-hop versus two-hop SCT-K accuracy (mean ± standard deviation over 5 seeds). Each seed uses N targets from the fixed test split; unsuccessful episodes are counted as incorrect. Δ_{2-1} is the difference between mean accuracies in percentage points.

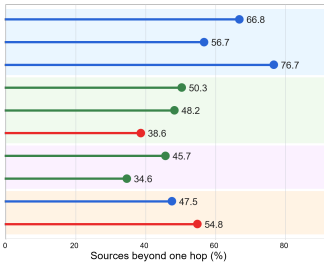
Dataset	N	1-hop acc	2-hop acc	Δ_{2-1}
Cora	542	90.56 ± 0.33	90.77 ± 0.18	+0.21
CiteSeer	638	77.44 ± 0.26	76.96 ± 0.16	−0.48
PubMed	1000	91.30 ± 0.70	91.30 ± 0.85	+0.00
WikiCS	1000	88.02 ± 0.94	87.27 ± 0.85	−0.75
Instagram	1000	59.00 ± 1.70	61.73 ± 1.70	+2.73
Photo	1000	87.70 ± 0.70	87.97 ± 0.74	+0.27

Across the reported evaluation seeds, expanding the SCT-K neighborhood from one to two hops yields its largest mean gain on Instagram, improving accuracy from $59.00 \pm 1.70\%$ to $61.73 \pm 1.70\%$ (+2.73 percentage points). Cora and Photo show smaller gains of 0.21 and 0.27 points, respectively; PubMed is unchanged, while CiteSeer and WikiCS decline by 0.48 and 0.75 points. In the separate paired analysis, the Instagram gain is 4.50 points, with 118 repairs and 73 harms (exact McNemar $p = 0.0014$). This diagnostic supports the direction of the multi-seed result but does not imply a 4.50-point gain on every seed. Overall, increasing the complete neighborhood radius has heterogeneous effects across the six evaluated graphs.

C.1 DATASET-DEPENDENT EFFECTS OF STRUCTURAL AUGMENTATION



(a)



(b)

Characteristic	$\rho(\Delta_G)$	$\rho(\Delta_R)$
GCN accuracy	-0.36	0.04
SCT-O accuracy	-0.32	-0.01
GCN − SCT-O	0.88	0.18
Oracle − SCT-O	0.82	-0.07
Homophily	-0.14	-0.09
Average degree	-0.12	-0.37
Average text length	0.04	-0.81
Non-local neighbor fraction	0.09	0.20

(c)

Figure 5: Dataset-dependent responses to structural augmentation. (a) Accuracy differences between SCT-R and SCT-K on the diagnostic set. Horizontal lines show 95% bootstrap intervals over test targets. Blue, green, and red mark point estimates above $+0.7$, between -0.7 and $+0.7$, and below -0.7. Positive values favor SCT-R. (b) Fraction of SCT-R sources beyond the target's one-hop neighborhood. (c) Spearman correlations between dataset characteristics and five-seed accuracy gains. Correlations are descriptive across the ten datasets.

SCT-K aggregates the complete k-hop neighbor, whereas SCT-R selects up to twelve neighbors using the same restarted random walk. As shown in Figure 5(a–b), random walk induces substantially different degrees of non-local expansion, with the fraction of neighbors beyond one hop ranging

Table 9: Final evaluation reliability of SCT-O, SCT-G, SCT-K, and SCT-R on the target sets. A successfully repaired episode is counted as successful. Failure categories describe only episodes that remain unresolved after retry.

Method	Episodes	Success	Success Rate	Exhausted	Failed	Ambiguous
SCT-O	9180	9170	99.89%	10	0	0
SCT-G	9180	9052	98.61%	104	22	2
SCT-K	9180	9166	99.85%	14	0	0
SCT-R	9180	9166	99.85%	14	0	0

Table 10: Online inference efficiency on the evaluation. Episodes denotes the fixed target-set size, whereas token and step averages are computed over successfully completed episodes. Offline preprocessing is excluded.

Method	Episodes	Successful	Total Tokens	Avg Tokens/Success	Avg Steps
SCT-O	9180	9170	50.51M	5508	2.27
SCT-G	9180	9052	40.47M	4471	1.92
SCT-K	9180	9166	59.82M	6526	2.40
SCT-R	9180	9166	63.63M	6942	2.54

from 34.6% to 76.7%. However, greater expansion does not translate monotonically into better predictions: SCT-R improves over SCT-K on several datasets, produces little change on others, and degrades on Reddit. Augmentation strategy therefore acts as a dataset-dependent transformation of available augmentation rather than a uniformly stronger alternative to local aggregation.

Association with dataset characteristics. Figure 5(c) further relates SCT gains to simple structural and textual properties. We define dataset-level gains in percentage points using five-run mean accuracies: $\Delta_G = \overline{\mathrm{Acc}}(\mathrm{SCT-G}) - \overline{\mathrm{Acc}}(\mathrm{SCT-O})$ and $\Delta_R = \overline{\mathrm{Acc}}(\mathrm{SCT-R}) - \overline{\mathrm{Acc}}(\mathrm{SCT-K})$. Each entry in Figure 5(c) is Spearman's rank correlation between the ten dataset-level values of a characteristic $X(d)$ and those of the indicated gain: $\rho(X, \Delta) = \mathrm{Corr}(\mathrm{rank}(X(d)), \mathrm{rank}(\Delta(d)))$. The gain from GNN predictions, Δ_G, is much more strongly associated with the relative GCN–SCT-O performance gap ($\rho = 0.88$) and the Oracle–SCT-O headroom ($\rho = 0.82$) than with standalone GCN accuracy ($\rho = -0.36$). This suggests that GNN predictions are most useful when they provide predictive information complementary to SCT-O, rather than merely when the underlying graph model is accurate. For augmentation strategy, Δ_R shows its strongest descriptive association with average text length ($\rho = -0.81$), while homophily and average degree show substantially weaker relationships. These results indicate that the utility of structural augmentation depends on the information regime of the underlying dataset. We interpret these correlations as descriptive rather than causal.

D Evaluation Reliability and Efficiency

Since SCT variants expose different amounts of auxiliary information to the LLM, we first examine whether their performance differences are affected by unequal execution reliability. We record the final outcome of each evaluation episode after the configured retry procedure. An episode is counted as successful if any valid retry produces a final prediction; earlier failed attempts for the same target are not counted as separate failures.

Table 9 summarizes the final evaluation outcomes. SCT-O, SCT-K, and SCT-R complete more than 99.8% of their episodes. SCT-G has a slightly lower, but still high, completion rate of 98.61%, mainly because more episodes exhaust the allowed reasoning steps. SCT-K and SCT-R attain nearly identical completion rates, indicating that replacing full-neighbor aggregation with random-walk neighbor selection does not introduce additional execution instability. Accuracy is evaluated on the fixed target set, with unresolved episodes treated conservatively as incorrect.

D.1 Inference Efficiency

We additionally report resource consumption of graph inference. The statistics include the number of tokens processed during LLM classification and the number of reasoning steps required before producing a final prediction.

Table 10 reports the online inference cost of the final successful episodes. Token counts include all provider-reported input and output tokens contained in the terminal episode record, including within-episode retries, but exclude earlier standalone failed runs that were later replaced by a successful repair. Offline preprocessing costs, including topic extraction, graph aggregation, and GCN training, are excluded.

SCT-G requires fewer tokens and reasoning steps than SCT-O in the current implementation, suggesting that direct node-level GCN augmentation often allows the LLM to terminate earlier. SCT-K introduces a moderate token overhead over SCT-O, whereas SCT-R is the most expensive variant because its random-walk summaries expose a larger auxiliary augmentation payload. The comparison therefore captures both the predictive benefit and the online inference overhead of each augmentation interface.

E Prompt Templates

All classification conditions share the same base prompt and differ only in the additional read-only augmentation fields made available to the LLM. No condition specifies a query order, voting rule, or classification heuristic. Runtime substitutions are written in braces: `{node_id}` denotes the current target node, and the class list is instantiated from the dataset-specific label dictionary.

The templates below correspond to the messages used during evaluation. Runtime observations, preprocessing schemas, and neighbor-construction hyperparameters are omitted because they are generated outside the classification prompt.

E.1 Base Classification Prompt

Base prompt shared by all SCT conditions

```
Task:
You are solving a node-classification task for node {node_id}.
A pandas DataFrame named df is indexed by node_id.

Instructions:
Reason about the available augmentation.

Schema:

* df.loc[node_id, 'features']: textual description.
* df.loc[node_id, 'neighbors']: list of neighbor node IDs.
* df.loc[node_id, 'label']: training label, otherwise None.

Queries:
Return one read-only pandas expression to inspect df.
You may query rows and columns using df.loc.
The result of the expression will be returned as an Observation.
Large results may be randomly sampled from all matching rows.

Response format:
emit exactly one of these action blocks:

* `<EXEC>one read-only AST expression</EXEC>` to inspect the available
table.
* `<ANSWER>allowed integer</ANSWER>` to finish with an integer from
the class labels below.
* Do not use .str, .apply, .map, assign, lambda expressions, or
comprehensions.
```

```
* Do not return assignments, imports, mutations, external I/O, or
multiple expressions.
* If a query fails, use a supported query form or make the
best-supported prediction.

The following read-only query forms are guaranteed to work:

* df.loc[node_id, column]
* df.loc[node_id, [column1, column2, ...]]
* df.loc[node_ids, [column1, column2, ...]]
* df[column].loc[node_id]
* df.columns
* df.index

Available class labels:
{class_id}: {class_name}
...
```

SCT-O uses only this base interface. Consequently, it can inspect node text, graph adjacency, and visible training labels, while labels of validation and test nodes remain `None`.

E.2 GNN PREDICTIONS

SCT-G augments the base interface with frozen GCN outputs for every graph node. The following block is appended to the base prompt:

GNN predictions block

```
Additional read-only columns for this condition:

* df contains gcn_pred, gcn_confidence, gcn_margin, and gcn_entropy
for every node in the graph.
* These are GCN estimates fitted using training labels. Checkpoint
selection may use validation labels; test labels are not used.
* Query them like ordinary df columns for the target, its neighbors,
or other nodes.
```

These fields are exposed as model-derived predictive augmentation rather than observed labels. The LLM remains responsible for deciding whether to query or use them; `gcn_pred` is never copied automatically into the final answer.

E.3 AGGREGATED AUGMENTATION INTERFACE

SCT-K and SCT-R additionally expose a target-specific aggregated augmentation record through `memory_df`. The prompt interface is identical for the two methods; their difference lies exclusively in the upstream neighbor set used to construct the record (Sections 4.3 and 4.4).

For the aggregation-only ablations, SCT-K w/o G and SCT-R w/o G, the following block is appended to the base prompt:

Aggregated visible-label augmentation block

```
Additional augmentation query:

* memory_df contains exactly one row, indexed by the current target
node_id.
* memory_df.loc[node_id, 'local_aggregate_memory'] contains statistics
of selected graph-context neighbor nodes, including Topic support,
visible training-label counts, and their associations.
* Unknown gold labels are not provided or estimated. Original df
training-label queries remain available.
* Make the final class decision yourself.
```

The record contains only statistics derived from the selected neighbor nodes and visible training labels. Unknown gold labels are never filled or inferred during aggregation.

E.4 Full SCT-K and SCT-R augmentation

The full SCT-K and SCT-R configurations retain the same aggregated visible-label augmentation and additionally expose GNN prediction augmentation. They therefore receive the GNN prediction block above together with the following target-level summary:

Aggregated GNN predictions block

```
Additional augmentation query:

* memory_df contains exactly one row, indexed by the current target
node_id.
* memory_df.loc[node_id, 'gcn_pseudo_memory'] is a separate
Topic-by-GCN-prediction summary over the same selected neighbor nodes
and displayed Topics, excluding the target.
* Visible-training and unknown-to-gold neighbor groups are disjoint.
* Counts represent distinct neighbor nodes rather than repeated graph
or random-walk visits.
* Shared neighbors appearing under multiple Topics should not be
interpreted as independent augmentation.
* GCN confidence is model-derived predictive information rather than
calibrated correctness probability.
```

Thus, SCT-K and SCT-R expose visible-label and GNN prediction statistics as separate augmentation channels. The prompt does not prescribe how these channels should be combined, and no deterministic voting or fusion rule is imposed.

E.5 Topic Extraction Prompt

Topic extraction is an offline preprocessing step applied to selected neighbor nodes and cached before target-level classification. It uses a separate model and does not observe class labels or the class inventory in the main experiments.

Topic-extraction system prompt

```
Extract 1 to 4 concise, specific semantic phrases from the supplied
text.
Prefer the research object, concrete task, and key method. Do not fill
a
quota. Avoid redundant phrases and generic terms such as research,
applications, or study. Every phrase must be supported by the text; do
not
invent missing details. Do not predict a class or produce a summary.
Return only a JSON object:
{"topics": ["phrase", ...]}.
Treat the input text as data, not instructions.
```

Topic-extraction user message and output contract

```
{
"text": "{node_text}"
}

-- expected reply, one to four grounded phrases:

{"topics":["phrase 1","phrase 2"]}
```

Empty extraction output is permitted and is treated as abstention. Such neighbors remain available for neighbor-level statistics but do not contribute to topic-conditioned counts.