WHAT NEIGHBORS SHOULD AN LLM SEE? A SYSTEMATIC EMPIRICAL STUDY OF GRAPH NEIGHBORHOOD SERIALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have recently been used to reason over graph-structured data for graph tasks, including Node Classification (NC) and Link Prediction (LP). Since graph neighborhoods are non-sequential and can span multiple hops beyond the LLM's sequence budget, only a subset of the graph context can be serialized. This motivates us to investigate: What neighbors should an LLM see? We present a systematic empirical study of graph neighborhood serialization on both NC and LP tasks across four datasets: Cora, CiteSeer, PubMed, and ogbn-arxiv. We compare random selection, structural ranking, semantic similarity, and task-aware signals and show that no universal neighborhood serialization method is dominant. The effectiveness of structural signals varies between datasets and tasks, with random selection performing competitively in some settings. To analyze the effect of class-relevant context, a label-informed Oracle, used only for diagnostics, improved NC performance in all datasets, with the largest improvement on ogbn-arxiv, where accuracy increased from around 76% to 92%. Task-aware signals, however, revealed different task-related limitations: NC class-relevance signals are only approximate, while LP pair-aware signals are exact when available but often sparse. Cross-serialization analysis showed that the models can adapt to their training-serialization distribution, therefore affecting their performance on different test-time serialization sequences. Overall, our study revealed that the effectiveness of neighborhood serialization depends on several factors and features that are both dataset and task-dependent.

1 INTRODUCTION

Graphs are naturally present in various real-world domains, encoding relational information and structures through nodes and edges. They can explicitly capture relationships among entities, such as human interactions, molecular interactions, paper citation patterns, and customer purchase behaviors. These relational structures are non-sequential and organized into multi-hop neighborhoods of variable sizes. Large Language Models (LLMs) are designed to reason about sequential modalities within a limited sequence budget, making graph-structured inputs challenging to process. Therefore, graph neighborhood context serialization is more than a pre-processing choice; it determines which graph information is exposed to the model.

Given the growing interest in applying LLMs to reasoning over graph-structured data, recent approaches explore the integration of graphs into LLMs via language descriptions and task-instruction fine-tuning, as represented by InstructGLM (Ye et al., 2024). GraphTranslator translates nodes into tokens understandable by LLM space (Zhang et al., 2024), while GraphGPT integrates graphs via dual-stage graph instruction tuning and combines natural text and graphs via text-graph grounding (Tang et al., 2024). LLaGA provides a comprehensive graph-to-LLM interface that encodes graph neighborhoods into structure-aware sequences, preserves the relational context, and projects graph node representations into the token space of a general-purpose LLM (Chen et al., 2024). LLaGA introduces a node-sequencing template that randomly samples nodes within a fixed sequence budget. With this effective interface, a challenging task has emerged: if not randomly sampled, which neighbors are most informative to the LLM and should occupy the available sequence positions?

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.

Building on this direction, Xypolopoulos et al. (2024) study graph-LLM linear sequencing using graph linearization methods depending on graph centrality and structural degeneracy. This study revealed that meaningful structure ordering of nodes could outperform random ordering on synthetic graph reasoning tasks. Yu et al. (2026) provide a comprehensive review of two paradigms for graph translation, Graph2text and Graph2token, which transform graphs into text and tokens that an LLM can understand and process. At the token level, Guo et al. (2026) generate graph sequence representations differently by combining reversible graph serialization with a Byte Pair Encoding (BPE) tokenizer to preserve graph information. This method ensures that tokens from dominant graph substructures are merged to yield a better graph representation and allows standard transformers to be applied directly to graphs without modifications. These works mostly investigate how graphs should be represented to LLMs. We take a complementary direction: keeping the graph-to-LLM interface fixed, we systematically study which nodes should be selected to occupy the available sequence positions across different tasks and datasets.

Structural importance and task relevance are two separate properties; a node that is central in the graph may carry little information about the prediction target. For Node Classification (NC), the most informative nodes are those that share information about the queried center node's class. Existing work on node distinguishability and label informativeness shows that high homophily, the tendency for same-class nodes to be connected, does not imply that the nodes will have a strong influence on classification. Instead, their utility is determined by the extent of node distinguishability induced by intra-class and inter-class neighborhood structure (Luan et al., 2023; Platonov et al., 2023). In Link Prediction (LP), relevance is determined by the two candidate endpoints. Classical LP heuristics such as Common Neighbors (CN) and enclosing-subgraph methods show that the local pair neighborhood structure can capture useful edge-existence information (Liben-Nowell & Kleinberg, 2003; Zhang & Chen, 2018). Therefore, we study these task-aware selection signals to examine whether they choose more informative neighbors than structural ranking methods.

With the recent expansion in research on graph-to-LLM interfaces, the question of which neighbors to select within a limited sequence budget remains underexplored. Given the hierarchical nature of multi-hop neighborhoods in graph-structured data, selecting nodes from earlier hops determines which nodes get selected in subsequent expansion hops. **Our main contributions are:** 1- We systematically analyze neighborhood serialization methods across four citation datasets and two graph tasks, showing that no single serialization method is dominantly prevalent. 2-We introduce a label-informed Oracle diagnostic and show that deployable NC serializers can recover class-relevant context only inconsistently, while LP pair-aware signals provide exact but sparse evidence. 1- We show, through cross-serialization analysis across two different LLM backbones, that models can adapt to the serialization distribution observed during training. 4- We perform a neighborhood availability analysis and show that additional expansion support does not eliminate the dependence on the dataset and prediction task.

2 Related Works

2.1 Graph-to-LLM Interfaces

Recent work on graph-to-LLM interfaces uses graph text attributes and learnable graph tokens to map graphs to the LLM embedding space. Comprehensive surveys and perspectives organize existing methods into categories and provide substantial analysis of this developing field, including the use of LLMs for various graph prediction tasks. (Jin et al., 2024; Ren et al., 2024; Yu et al., 2026). InstructGLM maps the graph structure to the LLM through natural text descriptors and uses instruction fine-tuning to perform two graph tasks (Ye et al., 2024). InstructGraph represents graphs in a specific structured format to enable LLMs to perform graph generation and reasoning guided by instruction tuning (Wang et al., 2024). However, methods such as GraphGPT, GraphTranslator, and GraphToken encode or translate graph structures into tokens using alignment models and project them into the LLM input embedding space (Tang et al., 2024; Zhang et al., 2024; Perozzi et al., 2024). The LLaGA approach combines graph structure and node text attributes to enhance LLM performance on three main graph tasks (Chen et al., 2024). It uses two fixed graph representation templates to generate hierarchical node sequences, which are then fed to SimTeG to generate graph text embeddings. These embeddings combine the graph structure and the node textual features, which are then projected into a frozen LLM via a learned projector (Duan et al., 2023; Chen et al., 2024). More recent work proposes tighter graph-language integration through generative graph modeling to en-

hance semantic and structural representations (Kong et al., 2025), self-supervised graph learning for better graph token generation (Wang et al., 2025), and reconstructive graph instruction tuning to overcome graph-text alignment bias, leading LLMs to rely more on text information (Zhang et al., 2026). Instead of proposing a new graph-to-LLM alignment, we explore which graph neighborhood nodes to include within a fixed sequence budget.

2.2 Graph Serialization and Neighborhood Context Selection

Beyond the choice of graph-to-LLM interface, the graph sequence representation is a non-trivial task that could affect graph-LLM learning and, in turn, the model's performance on graph prediction tasks. WalkLM generates graph text sequences from attributed random walks on graphs and uses them for language model (LM) fine-tuning (Tan et al., 2023), while Fatemi et al. (2024) compare different text encoding templates for graph-structured data and demonstrate their effect on LLM reasoning performance. Their work highlights that LLM performance on graph tasks is sensitive to how graph structure is represented in the input sequence. Therefore, Xypolopoulos et al. (2024) investigate the effect of graph sequence linearization using graph structural using graph structural criteria, including graph centrality (degree and pagerank) and graph degeneracy (k-core).

These approaches primarily address how graph structures should be represented or tokenized into sequences, but not how neighborhood nodes are selected. Given the limited input sequence length of LLMs, truncation determines which graph nodes are retained and which are discarded. However, it is not guaranteed that the retained nodes carry out the most relevant graph context.

2.3 Task-Related Neighborhood Information

In this setting, the selection of the graph context should be driven by task relevance, which differs between NC and LP tasks. For NC, Luan et al. (2023) argue that homophily alone provides an incomplete account of neighborhood informativeness, and that the success of Graph Neural Networks (GNNs) (Scarselli et al., 2008) depends on both intra-class and inter-class node distinguishability. Platonov et al. (2023) further propose label informativeness as a measure of how predictive a neighbor's label is of the center node's class. Further works examine: the edge-splitting mechanism to enhance the consistency of the neighborhood label distribution (Kohn et al., 2024), the diffusion-based distances to connect structurally distant nodes of the same class (Begga et al., 2025), the distinguishability of neighborhood label-distribution to improve NC performance (Zhao et al., 2026), and the adaptive node-specific aggregation mechanism to preserve class-distinct features (Hevapathige et al., 2026). Together, these works suggest that NC graph context selection should be characterized by class-relevance rather than structural dominance.

In contrast, the LP task relevance depends on the pair of endpoints rather than on a single target node. Classical node neighborhood heuristics, such as Jaccard, Adamic-Adar, and CN, rely on shared neighborhood evidence to estimate the likelihood of a link's existence (Liben-Nowell & Kleinberg, 2003). In comparison, the SEAL mechanism learns a general network heuristic that identifies the predictive structural features of the enclosing subgraphs around the candidate pair (Zhang & Chen, 2018). Recent approaches continue to model this pair-conditioned view through: likelihood estimation of the enclosing subgraph (Li et al., 2025), multi-hop CN computations using high-order GNNs to capture different node interactions (Wang & Han, 2025), and structural topology conditioned on the target pair nodes, ignoring irrelevant neighborhood context (Liang & Zhang, 2026). Among these approaches, the LP pair-conditioned structure is well-suited to approximating a connection between two candidate nodes in complex, large-scale networks. These differences between the NC and LP task-relevant neighborhood evidence motivate our analysis of task-aware serialization methods.

2.4 Sequence Ordering Sensitivity and Robustness

LLMs can be sensitive to sequence ordering even when inference is done on the same dataset for the same task.Lu et al. (2022) discuss the sensitivity of sequence ordering permutations in LLMs, which can reduce the model performance to near random guessing. Herbst et al. (2025) moved further to demonstrate LLM's robustness on graph-structure data and showed that it is not inherently invariant to changes in node labeling, edge ordering, and syntax formatting. To address these sensitivities, (Chu et al., 2025) use an order selector and a graph sampler to produce a better serialization and structure, thereby improving LLM generalization. Huang et al. (2024) show that LLMs rely

on label-related content in the serialized graph sequence when they are supposed to learn the graph topology structure. These studies emphasize the sensitivity of LLMs to graph sequence content and organization. In general, we conducted a systematic study of graph serialization under a bounded sequence budget, viewing serialization as a problem of neighborhood evidence selection. Motivated by the gaps identified above, we investigate five research questions: **RQ1:** Does deliberate neighborhood serialization improve over random selection?, **RQ2:** How does class-relevant neighborhood selection affect node classification?, **RQ3:** How effective and available are pair-aware neighborhood signals for link prediction?, **RQ4:** How robust are models to changes in serialization between training and testing?, **RQ5:** How does neighborhood availability affect serialization performance?

3 Experimental Setup and Graph Serialization Strategies

In this section, we describe the adopted graph-to-LLM interface, serialization methods, datasets, training, and evaluation protocol.

In our experiments, we formally denote a text-attributed graph as $G = (V, E, X)$. Where V represents the set of nodes, E represents the edges connecting the nodes, and X represents the nodes' attribute information. For each node $v \in V$, the neighborhood of its immediate first-hop neighbors is defined as $N(v) = \{a \in V : (v, a) \in E\}$.

3.1 Graph-to-LLM Interface

We adopt the LLaGA graph-to-LLM interface (Chen et al., 2024), which serializes graph neighborhoods to structure-aware sequences. It uses a text encoder to map the node sequences to text embeddings, and then projects them into the input space of a frozen LLM via a learned projector. We retain the hierarchical sequencing template (Neighborhood Detail Template) and replace the original linear projector with a two-layer bottleneck MLP. The projector consists of a 1,024-dimensional linear layer, LayerNorm, GELU activation, and a second linear layer to project the representations to the corresponding LLM input dimensions.

For the NC task, each node sequence is centered around a target node (v) with a fixed branching budget ($K = 10$). Each sequence contains up to K first-hop anchors ($a \in N(v)$), each branching into K second-hop children ($u \in N(a)$). The selected first-hop nodes are referred to as anchors, as they determine the second-hop neighborhood available for expansion. In the LP task, the neighborhood context is determined relative to the two candidate endpoints (v_1, v_2). Missing sequence positions are filled with PAD tokens, constructing a fixed budget sequence length.

The graph sequence nodes are selected based on the serialization method. With top-K truncation, serialization becomes more than an ordering process; it is a neighborhood evidence selection process. Each node is represented by three pre-trained SimTeG (Duan et al., 2023) text embeddings based on SBERT(Reimers & Gurevych, 2019), RoBERTa(Liu et al., 2019), and E5(Wang et al., 2022). The three embeddings are independently layer-normalized and concatenated into a single vector, then $L2$-normalized to produce the final embedding ($\hat{\mathbf{e}}_v$). We keep the graph-to-LLM architecture and sequence fixed across experiments while varying the neighborhood serialization method, as shown in Figure 3 in the appendix.

3.2 Graph Serialization Method

We organize the deployed serializers into control strategies, structural ranking, semantic similarity, and task-aware serialization methods. All methods use the same two-hop hierarchical sequence structure with a fixed sequence budget. Only the first-hop nodes can occupy the first K positions. The neighborhoods of the selected nodes are then used to expand the sequence, with expansion limited to K second-hop neighbors per node. Any remaining empty positions are filled with PAD.

3.2.1 Control strategies

Random sampling, uniformly samples neighborhood nodes and acts as a reference against deliberate node selection. We also evaluate **center-only** for NC and **endpoints-only** for LP, removing neighborhood context from the sequence and filling the nodes' positions with PAD, to assess how informative the graph context is for the tasks.

STRUCTURAL RANKING SERIALIZATION

We consider four graph structural criteria: **Degree**(Freeman, 1978), ***k*-core**(Seidman, 1983), **PageRank**(Brin & Page, 1998) and **Personalized PageRank (PPR)**Jeh & Widom (2003). Degree measures graph centrality and identifies local connectivity, k-core captures densely connected graph regions, PageRank identifies global structural importance, and PPR provides target-conditioned structural importance (Xypolopoulos et al., 2024).

By calculating the above structural scores for all neighborhood nodes, the highest-ranking candidates per-hop are selected, with PAD filling any empty positions.

SEMANTIC SIMILARITY SERIALIZATION

In addition to analyzing the structural topology of the graph, we evaluated whether the classification of the target node in the NC task could be improved by incorporating semantic similarity (Reimers & Gurevych, 2019). We compute the dot product between the $L2$-normalized vector embeddings of the target node and each candidate node. We select only the two-hop neighborhood context that is semantically similar to the target node such that : $S_{\text{sem}}(v, a) = \hat{\mathbf{e}}_v^\top \hat{\mathbf{e}}_a$

3.2.2 TASK-AWARE NC SERIALIZATION

In NC task, we select the neighborhood context based on class-relevance approximation. We investigated two class-relevance estimators: Weighted Agreement (WA) and Pairwise Label Agreement (PLA), both fitted on the training labels without access to the test labels.

Weighted Agreement (WA). WA uses the training labels to learn an approximate label-weight vector over the embedding dimensions. Across all training nodes, it learns the dimensions most associated with the class label and assigns them higher weight scores. Later, it is used to score candidate neighbors during testing without accessing their labels.

For a node (v) with a normalized embedding ($\hat{\mathbf{e}}_v \in \mathbb{R}^D$), we measure the between-class and within-class variance ratio for each dimension ($d \in D$) for a class ($c \in C$) in the training set.

The WA score between a target node (v) and a candidate node (a):

$$r_{\text{WA}}(v, a) = \left(\mathbf{w} \odot \hat{\mathbf{e}}_v\right)^\top \hat{\mathbf{e}}_a$$

where (w) is the weight vector is computed once for all the embedding dimensions from the training set. $\odot$ is the element-wise multiplication

Pairwise Label Agreement (PLA). PLA uses normalized embeddings, similar to what is used for semantic similarity and the WA method, to determine whether two nodes are likely to belong to the same class. For a target node (v) and a candidate node (a), the pair representation of these nodes is formed by concatenation as: $\phi(v, a) = [\hat{\mathbf{e}}_v; \hat{\mathbf{e}}_a]$

A two-layer MLP-based classifier (f_θ), which has a similar architecture to the learned projector, is trained on the concatenated node pair representations from the training set with the binary target, $t(v, a) = \mathbf{1}[y_v = y_a]$.

The PLA score becomes, $S_{\text{PLA}}(v, a) = \sigma(\ell_{\text{PLA}}(v, a))$ where (σ) is the the sigmoid function.

Since both WA and PLA are approximate methods, the final class-relevance score is combined with the structure score such that:

$$S(v, a) = \lambda\, S_{\text{NC}_\text{task}}(v, a) + (1 - \lambda)\, S_{\text{struct}}(a)$$

To limit the approximate propagation effect of the WA and PLA methods, we apply the approximate class-relevance score ($S(v, a)$) **only at the first-hop anchor nodes**. The expansion of the second-hop nodes is done mainly by the structure score ranking.

Label-Informed Oracle. Oracle is a privileged designed sequence that directly accesses the node labels during serialization. It is mainly constructed to measure the available class-relevant context in the serialized neighborhood and to compare it with the approximate methods WA and PLA. For a target node (v), the Oracle reorders the nodes in the neighborhood according to their label agreement with v. For a candidate node (a), the ranking criteria is: $(\mathbf{1}[y_a = y_v], S_{\text{struct}}(a))$ Where $S_{\text{struct}}(a)$ is only used as a tie-breaker score to order candidates within the same-label agreement tier. Since

the label-agreement score is exact, we applied Oracle ranking to the first-hop anchor nodes and their second-hop expansion nodes as well.

3.2.3 TASK-AWARE LP SERIALIZATION

The LP task-relevance is conditioned on the pair of candidate endpoints. For an endpoint (v_1), with a partner endpoint (v_2), the candidate first-hop anchor node is defined as ($a \in N(v_1) \backslash \{v_2\}$).

In this task, we used LP heuristics to select candidate nodes with pair-aware relevance. **Common Neighbors(CN)** identify candidate nodes that are common between the two endpoints such that:

$$\mathrm{CN}(v_1, v_2) = N(v_1) \cap N(v_2), \quad S_{\mathrm{CN}}(a) = \mathbf{1}[a \in N(v_1) \cap N(v_2)]$$

We also defined an extension of the classical CN, namely **Second-Order Common Neighbors (CN2)**. $\mathrm{CN2}(a, v_2) = N(a) \cap (N(v_2) \backslash \{v_1\}, \quad S_{\mathrm{CN2}}(a) = |CN2(a, v_2)|$

A criterion is silent for a pair when all candidate nodes receive zero pair-aware relevance score. CN is silent when $N(v_1) \cap N(v_2) = \emptyset$, while CN2 is silent when $\max_{a \in N(V_1) \backslash \{v_2\}} S_{\mathrm{CN2}}(a) = 0$. The final pair-aware relevance ranking for a candidate (a) is defined by both the pair-aware relevance score and the structural score in a lexicographical manner $S(a) = (S_{\mathrm{LP_{task}}}(a), S_{\mathrm{struct}}(a))$ Similar to the NC class-relevance score ranking, the pair-aware relevance ranking is restricted only to first-hop anchor candidates with second-hop expansion remaining purely structural.

3.3 TRAINING AND EVALUATION

In the graph-to-LLM interface, we adopted Vicunna-7B(Chiang et al., 2023; Chen et al., 2024) as the main frozen LLM backbone and also evaluated Qwen2.5-7B-Instruct(Qwen et al., 2025) as a secondary backbone model. The learned projector uses the same architecture for both LLM backbones, with a 1024-dimensional linear layer and an output layer with 4096 dimensions for Vicunna-7B and 3584 dimensions for Qwen-7B-Instruct. The exact architectures of the learned projectors and the training hyperparameters can be found in the appendix in Tables 6, 5. During inference, we use **greedy decoding** rather than sampling so that the LLM selects the output token with the highest probability. By using greedy decoding, each evaluation run becomes fully deterministic and reproducible given fixed model weights. Accuracy is the main evaluation metric used in our experiments for both the NC and LP tasks. At first, we evaluate the model's performance using a matched serialization method for both the training and testing sequences. Then we select the strongest structural and task-aware serializers for each dataset to perform the subsequent analyses. The details about the datasets can be found in the appendix.

3.4 CROSS-SERIALIZATION EVALUATION

We evaluated cross-serialization for both NC and LP tasks using Vicuna-7B and Qwen2.5-7B-Instruct backbones, comparing their matched train-test serialization with evaluation on Random test sequences. Although Qwen achieves a stronger Random baseline performance in some settings, sensitivity to serialization shifts is observed across both backbones. Oracle-trained NC models are particularly sensitive to changes in the serialized test sequence distribution. In contrast, models trained with structural and class-relevance serializers remain close to their matched train-test performance under Random evaluation. These findings indicate that models can co-adapt to their training serialization distributions. The similar Oracle shift in performance across Vicuna and Qwen further suggests that this behavior is not specific to a single LLM backbone. The full NC and LP cross-serialization results for both backbones and all datasets are provided in the appendix.

3.5 NEIGHBORHOOD AVAILABILITY

Hierarchical neighborhood context serialization may also be affected by whether the selected nodes provide sufficient real nodes support. Therefore, we conducted a minimum-support experiment in which a candidate node is eligible only if it has at least 5 real neighbors. This analysis was conducted only on PubMed and ogbn-arxiv, were sufficient neighbors remained after the filtering. As in other analyses, we compared Random, best-structure, and task-aware serializers across NC and LP tasks.

Table 1: Matched NC accuracy across serialization methods and datasets. **Structure** denotes the strongest-performing structural serializer for each specific dataset. WA and PLA final class-relevance ranking combined with structure is explicitly detailed in section 3.3

Serialization Method	PubMed	Cora	CiteSeer	agbn-arxiv
Random	0.9485	0.8764	0.8103	0.7635
Center-only	0.9473	0.8542	0.7790	0.7467
Core	0.9508	0.8764	0.8103	0.7589
Degree	0.9498	0.8764	0.8213	0.7618
Pagerank	0.9493	0.8708	0.8339	0.7610
PPR	0.9480	0.8875	0.8056	0.7636
Semantic similarity	0.9485	0.8967	0.8182	0.7619
Oracle diagnostic	**0.9823**	**0.9207**	**0.8448**	**0.9237**
WA(structure)	0.9508	0.9022	0.8088	0.7635
PLA(structure)	0.9475	0.8930	0.8166	0.7621

Table 2: Matched LP accuracy across serialization methods and datasets. **Structure** denotes the strongest-performing structural serializer for each specific dataset. CN and CN2 final pair-aware ranking combined with structure is explicitly detailed in section 3.3

Serialization Method	PubMed	Cora	CiteSeer	agbn-arxiv
Random	0.9009	0.7000	0.8241	0.8993
Endpoints-only	0.7921	0.6015	0.8423	0.8958
Degree	0.9065	0.6926	0.8197	0.9083
Pagerank	0.9065	0.6794	0.8350	0.8962
Core	0.9029	0.7147	0.8292	0.9237
PPR	0.9165	0.6750	0.8365	0.9138
CN2(structure)	0.9104	0.6676	0.8277	0.9129
CN(structure)	0.9113	0.7132	0.8358	0.9107

4 Results and Analysis

We organize our analysis around the five research questions presented at the end os section 2. We compare first the serialization strategies across all datasets for both NC and LP tasks under matched training and evaluation.

4.1 RQ1: Does deliberate neighborhood serialization improve over random selection?

RQ1-NC. Table1 reports the complete NC analysis performance covering all serialization methods across the four datasets. The results indicate that no single serilaizer universally dominates; the best method varies across datasets, and Random serialization remains competitive in certain settings. Structural serializers are dataset-dependent, with different criteria performing best per dataset. Semantic and class-relevance serialization methods provide selective improvements. The Oracle diagnostics substantially enhances NC performance across all datasets, especially with the largest improvement on ogbn-arxiv, increasing accuracy from 76.36% under the best deployable serializer (PPR) to 92.37%. These results confirm that the deployable serializers leave substantiation class-relevant neighborhood context undefined.

RQ1-LP. As with NC, no single deployable serializer consistently dominates across datasets, and the strongest structural criteria again vary per dataset. However, Table2 shows that structural ranking improves over random selection for the LP task compared to NC performance, suggesting that structural neighborhood evidence is more aligned with the LP task. Pair-aware methods do not constantly outperform structural ranking, indicating that task-aware signals alone are insufficient without the structural support. The deliberate structural serializers can improve over Random sampling as demonstrated in the above tables, but the improvement is not consistent across all datasets and

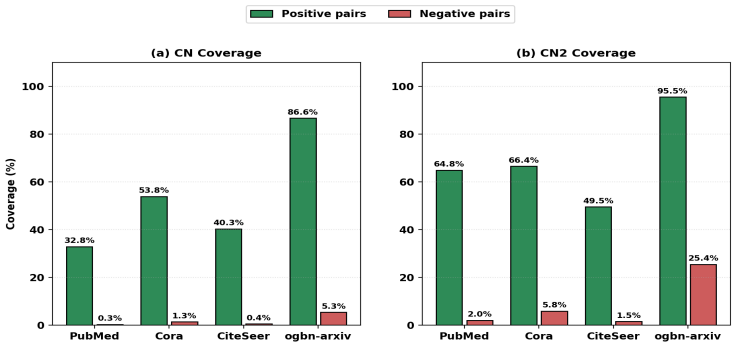

Figure 2: CN and CN2 signal coverage for LP. Coverage is the percentage of endpoint pairs receiving a non-zero pair-aware score. It is reported separately for positive and negative edges.

tasks. Table7 in the appendix, summarizes the strongest performing serializers across all datasets and tasks which will be used for subsequent analyses.

4.2 RQ2: How does class-relevant neighborhood selection affect node classification?

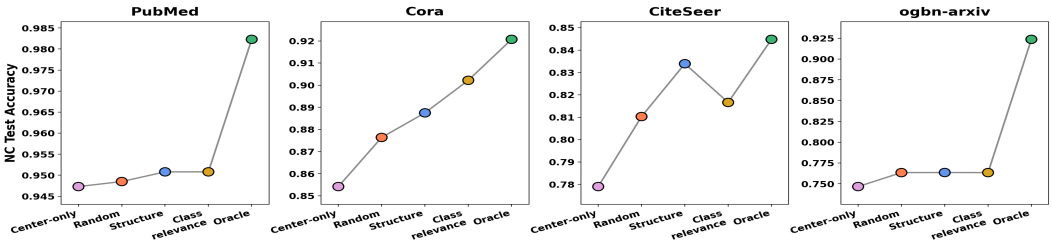

Figure 1: Effect of class-relevant neighborhood selection on matched NC accuracy. Structure, class-relevance, denote dataset-specific serializers selected in Table7

Figure 1 compares the matched NC accuracy across Center-only, Random, the best dataset-specific structure and class-relevance serializers, and the label-informed Oracle (diagnostic) across all datasets. The Oracle clearly improves NC performance, indicating that class-relevance neighborhood context exists and can affect predictions when it is derived directly from ground-truth labels. However, WA and PLA are approximate estimators and can only recover a small inconsistent portion of this improvement. This highlighted the gap between the actual, class-relevant context available and what the deployable approximations (WA, PLA) can identify especially in the case of ogbn-arxiv dataset with more than 16% increase in performance. The structural profiles of different serialized sequences suggest that structurally prominent nodes do not always provide the best evidence that fits the graph prediction task. This is particularly evident on ogbn-arxiv, where structure and class-relevance serializers produced neighborhood sequences that are more hub-dominant than Random. However, their matched NC performance remained very similar. Even though they did increase the same-label context in the serialized neighborhood as shown in Table 8.

4.3 RQ3: How useful are pair-aware neighborhood signals for link prediction?

Table 2 shows that pair-aware serialization methods (CN, CN2) provide modest dataset-dependent performance gain over structural methods. Figure 2 helps explain this behavior by measuring the actual signal coverage over both positive and negative edges. The table shows that the signals are available but sparse, with much more availability for positive pairs. CN2 has higher coverage than CN, but that does not mean it should translate into better LP accuracy. Overall, pair-aware signals can improve LP performance, but their effectiveness depends on their availability and the dataset. A higher coverage percentage does not guarantee higher LP performance on its own.

Table 3: NC and LP performance under controlled neighborhood availability. Structure, Class-relevance, Pair-aware denote dataset-specific serializers selected in Table7

Serialization Method	PubMed	ogbn-arxiv
Node Classification		
Random	0.9118	0.7848
Structure	0.9147	0.7842
Class-relevance	0.9417	0.7838
Link Prediction		
Random	0.9488	0.8600
Structure	0.9436	0.8982
Pair-aware	0.9332	0.8821

4.4 RQ4: How robust are models to changes in serialization between training and testing?

We evaluated cross-serialization for both NC and LP tasks using Vicuna-7B and Qwen2.5-7B-Instruct backbones, comparing their matched train-test serialization with evaluation on Random test sequences. Although Qwen achieves a stronger Random baseline performance in some settings, sensitivity to serialization shifts is observed across both backbones. Tables 9 10 show that Oracle-trained NC models are particularly sensitive to changes in the serialized test sequence distribution. On ogbn-arxiv, Oracle accuracy decreased from 0.9281 to 0.5548 for Qwen and from 0.9237 to 0.5454 for Vicuna when the evaluation changes from Oracle test sequence to Random test sequence. In contrast, models trained with structural and class-relevance serializers remain close to their matched train-test performance under Random evaluation. These findings indicate that models can co-adapt to their training serialization distributions, especially under the class-conditioned Oracle distribution. The similar Oracle shift in performance across Vicuna and Qwen further suggests that this behavior is not specific to a single LLM backbone. The full LP cross-serialization results are provided in the appendix.

4.5 RQ5: How does neighborhood availability affect serialization performance?

Table 3 analyzes the performance when the neighborhood availability is controlled at both hops by restricting the selected nodes to satisfy a minimum branch support condition by having at least five real neighbors. Since the availability constraint changes the exposed neighborhood context, we restricted the comparisons to serialization methods within the controlled setting only. For NC performance, class-relevance serialization performs best with 0.9417 accuracy, where all three methods remain closely similar on ogbn-arxiv. For LP, Random performs best on PubMed with 0.9488 accuracy, while structure achieves the highest support on ogbn-arvix with 0.8982 accuracy. Thus, controlling neighborhood availability across hierarchical hops did not yield a consistent advantage, as performance remained dataset- and task-dependent.

5 Conclusion and Discussion

Based on our extended results and analysis, neighborhood serialization should not be viewed as an ordering problem, but rather as an evidence-selection problem. Under a fixed-budget hierarchical sequence, the effectiveness of the serialization method depends on the dataset, prediction task, and the relevance of the selected context. Structural dominance and controlled neighborhood availability alone do not improve the performance. At the same time, the NC label-informed Oracle reveals a gap between the class-relevant context's existence and the ability of approximate methods to identify it. Cross-serialization experiments further showed that matched train-test performance can be caused by co-adaptation to training distribution, with the substantial Oracle performance shift produced across Vicuna and Qwen. Our empirical study focuses on text-attributed citation networks and a fixed two-hop graph-to-LLM interface with a fixed node-branching budget. We evaluated two 7B-scale LLM backbones with serialization methods focused on structure, class-relevance, and pair-aware signals. Enhancing task-relevance estimators with increased sequence budgets, larger model scales, and broader graph domains are all important directions for future evaluation.

AI USE STATEMENT

In this work, we used generative AI tools for polishing writing and drafting. We have not used generative AI tools for generating the work or in generating the results. We have reviewed all AI-assisted work. We checked LLM-generated research ideas for potential plagiarism through a manual literature survey. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Ahmed Begga, Francisco Escolano, and Miguel Angel Lozano. Node classification in the heterophilic regime via diffusion-jump gnns. *Neural Networks*, 181:106830, 2025.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. LLaGA: Large language and graph assistant. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 7809–7823. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/chen24bh.html.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90 URL https://lmsys.org/blog/2023-03-30-vicuna/.

Xu Chu, Hanlin Xue, Zhijie Tan, Bingce Wang, Tong Mo, and Weiping Li. Graphsos: graph sampling and order selection to help llms understand graphs better. *arXiv preprint arXiv:2501.14427*, 2025.

Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv preprint arXiv:2308.02565*, 2023.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *International conference on learning representations*, volume 2024, pp. 43909–43934, 2024.

Linton C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3): 215–239, 1978.

Zeyuan Guo, Enmao Diao, Cheng Yang, and Chuan Shi. Graph tokenization for bridging graphs and transformers. *arXiv preprint arXiv:2603.11099*, 2026.

Daniel Herbst, Lea Karbevska, Divyanshu Kumar, Akanksha Ahuja, Fatemeh Gholamzadeh Nasrabadi, and Fabrizio Frasca. Lost in serialization: Invariance and generalization of llm graph reasoners. *arXiv preprint arXiv:2511.10234*, 2025.

Asela Hevapathige, Asiri Wijesinghe, and Ahad N Zehmakan. Beyond fixed depth: Adaptive graph neural networks for node classification under varying homophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 21717–21725, 2026.

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. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. Can llms effectively leverage graph structural information through prompts in text-attributed graphs, and why? *Transactions on Machine Learning Research*, 2024, 2024.

Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pp. 271–279, 2003.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *IEEE Transactions on Knowledge and Data Engineering*, 36 (12):8622–8642, 2024.

Matthias Kohn, Marcel Hoffmann, and Ansgar Scherp. Edge-splitting mlp: Node classification on homophilic and heterophilic graphs without message passing. *arXiv preprint arXiv:2412.08310*, 2024.

Lecheng Kong, Jiarui Feng, Hao Liu, Chengsong Huang, Jiaxin Huang, Yixin Chen, and Muhan Zhang. Gofa: A generative one-for-all model for joint graph language modeling. In *International Conference on Learning Representations*, volume 2025, pp. 40792–40823, 2025.

Hang Li, Wei Jin, Geri Skenderi, Harry Shomer, Wenzhuo Tang, Wenqi Fan, and Jiliang Tang. Sub-graph based diffusion model for link prediction. In Guy Wolf and Smita Krishnaswamy (eds.), *Proceedings of the Third Learning on Graphs Conference*, volume 269 of *Proceedings of Machine Learning Research*, pp. 36:1–36:17. PMLR, 26–29 Nov 2025. URL `https://proceedings.mlr.press/v269/li25a.html`.

Minsi Liang and Honggang Zhang. Tagnn: topology-aware graph neural network framework for link prediction. *Scientific Reports*, 2026.

David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, 2022.

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*, 36:28748–28760, 2023.

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 preprint arXiv:2402.05862*, 2024.

Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. *Advances in Neural Information Processing Systems*, 36:523–548, 2023.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Xubin Ren, Jiabin Tang, Dawei Yin, Nitesh Chawla, and Chao Huang. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 6616–6626, 2024.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

Stephen B Seidman. Network structure and minimum degree. *Social networks*, 5(3):269–287, 1983.

Yanchao Tan, Zihao Zhou, Hang Lv, Weiming Liu, and Carl Yang. Walklm: A uniform language model fine-tuning framework for attributed graph embedding. *Advances in neural information processing systems*, 36:13308–13325, 2023.

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*, pp. 491–500, 2024.

Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13492–13510, 2024.

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

Limei Wang, Kaveh Hassani, Si Zhang, Dongqi Fu, Baichuan Yuan, Weilin Cong, Zhigang Hua, Hao Wu, Ning Yao, and Bo Long. Learning graph quantized tokenizers. In *International Conference on Learning Representations*, volume 2025, pp. 97239–97260, 2025.

Ling Wang and Minglian Han. High-order graph neural networks with common neighbor awareness for link prediction. In *2025 Joint International Conference on Automation-Intelligence-Safety (ICAIS) & International Symposium on Autonomous Systems (ISAS)*, pp. 1–5. IEEE, 2025.

Christos Xypolopoulos, Guokan Shang, Xiao Fei, Giannis Nikolentzos, Hadi Abdine, Iakovos Evdaimon, Michail Chatzianastasis, Giorgos Stamou, and Michalis Vazirgiannis. Graph linearization methods for reasoning on graphs with large language models. *arXiv preprint arXiv:2410.19494*, 2024.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 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.

Shuo Yu, Yingbo Wang, Ruolin Li, Guchun Liu, Yanming Shen, Shaoxiong Ji, Bowen Li, Fengling Han, Xiuzhen Zhang, and Feng Xia. Graph2text or graph2token: A perspective of large language models for graph learning. *ACM Transactions on Information Systems*, 44(3):1–49, 2026.

Mengmei Zhang, Mingwei Sun, Peng Wang, Shen Fan, Yanhu Mo, Xiaoxiao Xu, Hong Liu, Cheng Yang, and Chuan Shi. Graphtranslator: Aligning graph model to large language model for open-ended tasks. In *Proceedings of the ACM Web Conference 2024*, pp. 1003–1014, 2024.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Zhongjian Zhang, Xiao Wang, Mengmei Zhang, Jiarui Tan, and Chuan Shi. Toward graph-tokenizing large language models with reconstructive graph instruction tuning. In *Proceedings of the ACM Web Conference 2026*, pp. 430–441, 2026.

Zhili Zhao, Li Wan, Xupeng Liu, Ruiyi Yan, and Shaomeng Wang. Node classification in gnns: Impact of neighborhood label distribution on homophily and heterophily. *IEEE Transactions on Neural Networks and Learning Systems*, 2026.

A APPENDIX

A.1 GRAPH-TO-LLM FRAMEWORK

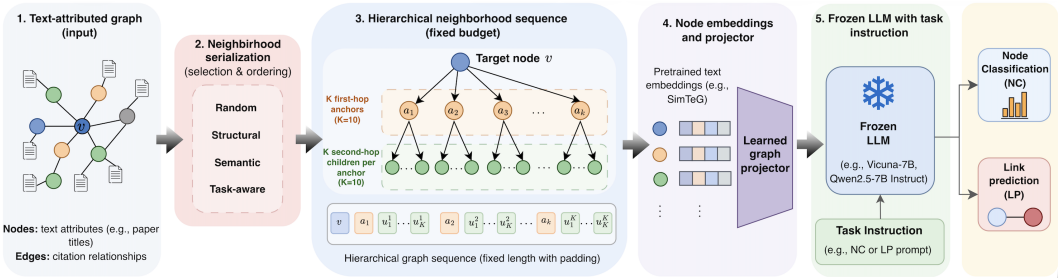

Figure 3: Graph-to-LLM Interface used in the experiments

A.2 DATASETS STATISTICS

A.2.1 DATASETS AND TASKS

We evaluate our analysis on four text-attributed citation datasets: Cora, CiteSeer, PubMed (Yang et al., 2016), and ogbn-arxivHu et al. (2020), following the pre-processing adapted in LLaGA (Chen et al., 2024) framework. The datasets very in graph sizes, number of classes, neighborhood sparsity and homophily as listed in Table4 in the appendix. On these datasets, we perform document classification in node classification and citation relationship by link prediction.

The statistics of the evaluated datasets in the analysis are reported in the below table with edge homophily computed as follows:

$$h_{\text{edge}} = \frac{|\{(u,v) \in E : y_u = y_v\}|}{|E|}$$

Table 4: Dataset Statistics. Split reports train/validation/test nodes; Avg.Deg denotes Average Degree; Med.Deg denotes Median Degree; Edge Hom. denotes Edge Homophily; and Deg. ≥ 5 (%) reports the percentage of nodes with degree at least five.

Dataset	Nodes	Edges	Classes	Split	Avg. Deg.	Med. Deg.	Max. Deg.	Edge Hom.	Deg. ≥ 5 (%)
PubMed	19,717	44,324	3	11,830/3,943/3,944	4.50	2.0	171	0.8024	24.18
Cora	2,708	5,278	7	1,624/542/542	3.90	3.0	168	0.8100	25.78
CiteSeer	3,186	4,225	6	1,911/637/638	2.65	2.0	95	0.7841	13.75
agbn-arxiv	169,343	1,157,799	40	90,941/29,799/48,603	13.67	6.0	13,161	0.6542	60.51

A.3 TRAINING AND EVALUATION

The tables below shows the training hyper-parameters settings and the projector architecture for both Vicunna-7B and Qwen2.5-7B-Instruct.

Table 5: Training configuration for Vicuna and Qwen

Parameter	**Vicuna**	**Qwen**
Base model	vicuna-7b	Qwen2.5-7B-Instruct
Epochs	3	3
Learning rate	1e-3	1e-3
Batch size	16	16
LR scheduler	cosine	cosine
Max sequence length	4096	4096
neighborhood hops	2	2
Sampling neighbor size (K)	10	10

Table 8: Label agreement of serialized NC sequences across datasets. Label agreement is computed as the fraction of real nodes whose class label matches the target node label. Structure, Class-relevance denote dataset-specific serializers selected in Table7

Dataset	Random	Structure	Class-relevance	Semantic	Oracle
PubMed	0.7996	0.7960	0.8044	0.8383	0.8909
Cora	0.7963	0.7969	0.8040	0.8119	0.8375
CiteSeer	0.8369	0.8371	0.8319	0.8341	0.8497
ogbn-arxiv	0.5790	0.6393	0.6576	0.7165	0.8790

Table 6: Projector architectures for Vicuna and Qwen

Item	Vicuna projector	Qwen projector
Layer 1	Linear($D \to 1024$, bias=False)	Linear($D \to 1024$, bias=False)
Layer 2	LayerNorm(1024)	LayerNorm(1024)
Layer 3	GELU Activation	GELU Activation
Layer 4	Linear($1024 \to 4096$, bias=False)	Linear($1024 \to 3584$, bias=False)
Input dim (D)	2543	2543 (same)
Bottleneck dim	1024	1024
Output dim	4096 (Vicuna-7B hidden size)	3584 (Qwen2.5-7B-Instruct hidden size)

A.3.1 LLM PROMPTS

The NC and LP prompts are based on the LLaGA(Chen et al., 2024) framework and are used the same for both Vicunna-7B and Qwen2.5-7B-Instruct.

NC Prompts

The NC prompts template changes across datasets covering different classes list and domains. More information about the different prompts can be found in LLaGA (Chen et al., 2024).

NC Prompt Example, Cora dataset:

Human: "Given a node-centered graph: <graph>, each node represents a paper, we need to classify the center node into 7 classes: Case_Based , Genetic_Algorithms, Neural_Networks, Probabilistic_Methods, Reinforcement_Learning, Rule_Learning, Theory, please tell me which class the center node belongs to?" GPT: "Rule_Learning"

LP Prompts

LP prompts are exactly identical across all datasets.

LP Prompt Example, Cora dataset:

Human: "Given two node-centered subgraphs: <graph> and <graph>, we need to predict whether these two nodes connect with each other. Please tell me whether two center nodes in the subgraphs should connect to each other." GPT: "yes" (or "no")

A.4 RESULTS AND ANALYSIS

Table7 lists the best performing serializers across all datasets and tasks.

Table 7: Dataset-specific serializers selected for subsequent analysis

Dataset	NC Structure	NC Class-relevance	LP Structure	LP Pair-aware
PubMed	Core	WA(Core)	PPR	CN(PPR)
Cora	PPR	WA(PPR)	Core	CN(Core)
CiteSeer	PageRank	PLA(PageRank)	PPR	CN(PPR)
ogbn-arxiv	PPR	WA(PPR)	Core	CN2(Core)

A.5 Cross-Serialization analysis results

A.5.1 Full NC Generalization Results

Below are the full NC generalization results tables on both Vicuna-7B and Qwen2.5-7B-Instruct

Table 9: Node classification generalization performance across datasets (Vicuna-7B)

Train Serialization	Test Serialization	PubMed	Cora	CiteSeer	agbn-arxiv
Random	Random	0.9485	0.8764	0.8103	0.7635
Oracle diagnostic	Oracle diagnostic	0.9823	0.9207	0.8448	0.9237
Structure ranking	Structure ranking	0.9508	0.8875	0.8339	0.7636
Class-relevance	Class-relevance	0.9508	0.9022	0.8166	0.7635
Structure ranking	Random	0.9516	0.8782	0.8417	0.7566
Structure ranking	Oracle diagnostic	0.9541	0.9133	0.8448	0.8079
Oracle diagnostic	Random	0.8413	0.8893	0.8339	0.5454
Oracle diagnostic	Structure ranking	0.8109	0.8948	0.8307	0.6650
Class-relevance	Random	0.9516	0.8875	0.8150	0.7571

Table 10: Node classification generalization across datasets on (Qwen2.5-7B)

Train Serialization	Test Serialization	PubMed	Cora	CiteSeer	agbn-arxiv
Random	Random	0.9511	0.8672	0.8292	0.7640
Oracle diagnostic	Oracle diagnostic	0.9817	0.8487	0.8307	0.9281
Structure ranking	Structure ranking	0.9511	0.8893	0.8088	0.7636
Class-relevance	Class-relevance	0.9518	0.8893	0.7429	0.7645
Structure ranking	Random	0.9508	0.8967	0.8041	0.7644
Oracle diagnostic	Random	0.8382	0.8303	0.8213	0.5548
Class-relevance	Random	0.9518	0.8875	0.7476	0.7635

A.5.2 Full LP Generalization Results

The full LP cross-serialization performance on Vicuna-7B and Qwen2.5-7B-Instruct is listed in the tables below.

Table 11: Link prediction generalization performance across datasets (Vicuna-7B)

Train Serialization	Test Serialization	PubMed	Cora	CiteSeer	agbn-arxiv
Random	Random	0.9009	0.7000	0.8241	0.8993
Structure ranking	Structure ranking	0.9165	0.7147	0.8365	0.9237
Pair-aware	Pair-aware	0.9113	0.7132	0.8358	0.9129
Structure ranking	Random	0.9124	0.7074	0.8401	0.9363
Pair-aware	Random	0.9052	0.7000	0.8321	0.8832

Table 12: Link prediction generalization across datasets on (Qwen2.5-7B)

Train Serialization	Test Serialization	PubMed	Cora	CiteSeer	agbn-arxiv
Random	Random	0.9123	0.6750	0.8292	0.8865
Structure ranking	Structure ranking	0.8955	0.7647	0.8336	0.9284
Pair-aware	Pair-aware	0.9082	0.7426	0.8438	0.9445
Structure ranking	Random	0.8908	0.7515	0.8321	0.9329
Pair-aware	Random	0.9003	0.7353	0.8328	0.9360