Boosting with Fewer Tokens: Multi-Query Optimization for LLMs Using Node Text and Neighbor Cues

Yujie Fang[†], Xin Li[†*], Yuangang Pan[‡§], Xin Huang[¶], Ivor W. Tsang[‡§]
[†]School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
[‡]Centre for Frontier AI Research, Agency for Science, Technology and Research, Singapore
[§]Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore
[¶]Department of Computer Science, Hong Kong Baptist University, Hong Kong, China
[†]{fangyujie,xinli}@bit.edu.cn,[‡]{Pan_Yuangang, Ivor_Tsang}@cfar.a-star.edu.sg,[¶]xinhuang@comp.hkbu.edu.hk

***Abstract*—Recent studies have explored querying large language models (LLMs) to serve as predictors for graph mining tasks on text-attributed graphs (TAGs), establishing a promising paradigm that surpasses Graph Neural Networks (GNNs) in scalability and generalization. However, the high token costs of LLMs make this approach prohibitively expensive for large-scale node queries, and effective multi-query optimization solutions are currently lacking. By conducting information theory analysis at the single query level, we have gained insights that enabled the development of two multi-query optimization strategies: token pruning and query boosting. The token pruning strategy is designed to reduce token usage without compromising task performance by identifying saturated node queries and pruning tokens for these queries. Meanwhile, the query boosting strategy is designed to enhance task performance by enriching the context of unexecuted queries with pseudo-labels derived from previous queries through strategic scheduling, thereby maximizing the utility of these pseudo-labels. Extensive experiments applying these two strategies, either jointly or individually, to various existing methods demonstrate that the proposed approach serves our intentions well. Besides, this paper offers a fresh methodology for optimizing LLM processing of graph tasks, demonstrating great potential. For most natural graph data benchmarks in the field, it can save tokens by several orders of magnitude. For example, on the Ogbn-Products dataset, it could theoretically save up to 2×10^9 tokens.**

***Index Terms*—Multi-query optimization, graph mining, large language models**

I. Introduction

The ubiquitous presence of graphs in the real world has fueled extensive applications of graph mining across diverse industries, such as recommendation systems [1]–[3], fraud detection [4]–[6], and biological analysis [7]–[9]. Traditional graph mining methods primarily rely on Graph Neural Networks (GNNs); however, GNNs encounter significant challenges in practical deployments due to intrinsic limitations. Specifically, the feature aggregation of GNNs necessitates that the entire graph be fully known and processed collectively, leading to two challenges: (i) Resource Demands on Large Graphs [10], (ii) Inability to Handle Dynamic Nodes [11]. Additionally, GNNs necessitate a training phase, which poses two further challenges: (iii) Reliance on Labeled Data [12]. (iv) Non-generalizable learning: GNNs struggle to conduct inference on new graphs when their feature or label spaces differ from those of the training graph [13]. In recent years, numerous efforts have been made to improve GNN-based systems to address these challenges. These improvements include scalable GNNs designed to mitigate resource demands on large graphs [14], [15], inductive GNNs that enhance the handling of dynamic nodes [16], [17], and unsupervised GNNs that decrease reliance on labeled data [18], [19]. Despite these efforts, the challenge of Non-generalizable learning remains inadequately addressed, and few approaches are capable of effectively tackling the other three challenges simultaneously.

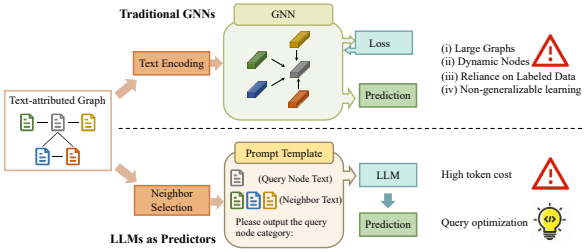

Fig. 1. Schematic diagram of GNNs (top) and LLMs as predictors (bottom), in the context of node classification tasks.

Recently, the remarkable abilities of large language models (LLMs) have positioned them as powerful tools in the field of data management. They are employed for various tasks including data enrichment [20], data transformation [21], and structured query language (SQL) generation [22]–[24], with the objective of improving efficiency, scalability, traceability, and beyond. For graph mining tasks, LLMs also offer new opportunities to tackle the previously discussed challenges. Recently, several studies [25]–[27] have investigated bypassing GNNs altogether, directly querying LLMs to serve as predictors in traditional graph mining tasks, a new paradigm termed "LLMs as predictors" [25]. These studies predominantly concentrate on node classification tasks on text-attributed graphs (TAGs), as the text attributes of each node serve as suitable

*Corresponding author.

2375-026X/25/$31.00 ©2025 IEEE
DOI 10.1109/ICDE65448.2025.00202

inputs for LLMs, and the task aligns well with the semantic comprehension strengths of LLMs. As illustrated in Fig. 1, each node to be classified has its neighbors selected to generate contextual text. This text, combined with the query node's information and a task description, forms a prompt that the LLM processes to classify the query node, thereby completing a single query. This paradigm is analogous to GNNs as both perform aggregation operations, with the aggregation targets being the neighbors' text and features, respectively. However, unlike GNNs, this paradigm handles each node's query independently and operates without the need for training, thereby circumventing the previously mentioned challenges: it can scale straightforwardly to large graphs, seamlessly handle dynamic nodes, reduce reliance on labeled data, and maintain versatility across multiple graphs.

While this paradigm holds great potential, deploying it to classify large numbers of nodes in real-world scenarios incurs substantial economic costs. For example, in work [25], each query from the citation network consumes at least 1,200 tokens, including the titles and abstracts of both the query article and five neighboring articles. Using even the cheapest version of GPT-3.5, priced at $0.0005/1k input tokens, a single query would cost at least $0.0006. Thus, handling a basic industrial-scale task with 10 million queries would cost at least $6,000, while using GPT-4 would increase the cost to $360,000. How to achieve optimal overall task performance for multiple queries under a limited token budget has become a critical issue, awaiting suitable multi-query optimization (MQO) solutions. However, most existing work in MQO [28]–[30] is oriented towards SQL queries, and the few studies [31]–[33] that do focus on LLM queries necessitate white-box access to LLMs, making them incompatible with the black-box LLMs commonly used in this paradigm. Additionally, these MQO studies rarely incorporate considerations for graph data scenarios. In response, this paper seeks to utilize the distinctive properties of graph data to develop MQO strategies tailored to optimizing the execution of multiple LLM queries.

Our approach begins by conducting an in-depth analysis at the single query level from an information theory perspective, seeking insights for optimizing multi-query execution. Considering that neighbor text is both a key design component of existing methods and the dominant token cost in each query, we identify it as pivotal in optimizing both cost and precision. By evaluating the information gain from neighbor text, we categorize nodes as saturated and non-saturated based on their potential to benefit from neighbor text: saturated nodes are those whose text attributes alone are adequate for LLMs to make accurate predictions, while non-saturated nodes need additional information from neighbor text. This guides MQO: reduce token costs by minimizing the neighbor text of saturated nodes and improve task performance by enriching the neighbor text of non-saturated nodes.

To meet token budget constraints, we propose a **token pruning** strategy designed to reduce token usage for multi-query execution within the "LLMs as predictors" paradigm. Unlike existing methods that uniformly equip neighbor text into prompts for all node queries, our token pruning strategy omits neighbor text for saturated node queries. Given that saturated nodes can be accurately predicted by LLMs with their own text, this strategy can maintain accuracy without sacrifice. To identify saturated nodes among multiple query nodes, we introduce a text-inadequacy measure that assigns smaller values to saturated nodes and larger values to non-saturated nodes. By sorting the text-inadequacy values of multiple queries in ascending order, users can flexibly determine the percentage of top-ranked queries to omit neighbor text, adapting to their token budgets. In essence, this strategy can provide a customizable token allocation solution that maximizes cost-efficiency tailored to each user's specific budget.

To optimize overall task performance, we propose a **query boosting** strategy designed to enrich neighbor text for multiple queries. The labels of neighbors are well-suited for augmenting neighbor text; these brief labels consume minimal tokens and generally benefit predictions, an advantage supported by the widely recognized homophily principle [34] that connected nodes often share similar labels. The fundamental idea of our query boosting strategy is to enrich a query' neighbor text with responses (pseudo-labels) from historical neighboring queries, building on the interconnected nature of multiple queries. In pursuit of optimal pseudo-label utilization, this strategy employs a query scheduling algorithm that arranges queries with fewer neighbor labels to be executed later, increasing their opportunities to integrate pseudo-labels from earlier executed queries. Overall, this strategy offers a cost-effective way to enrich the neighbor text of multiple queries.

Our main contributions are as follows:

- We propose a token pruning strategy that omits neighbor text for saturated nodes, aimed at reducing tokens without sacrificing accuracy. Furthermore, we propose identifying saturated nodes based on a text-inadequacy measure, providing a token allocation solution that optimizes cost-efficiency for users with varying token budgets.
- We propose a query boosting strategy aims to improve task performance by enriching the neighbor text of unexecuted queries with pseudo-labels from historical neighboring queries, and a query scheduling algorithm is used to maximize the utility of these pseudo-labels.
- Experimental results show that the token pruning strategy offers substantial token-saving potential and reduces token usage without harming accuracy, while the query boosting strategy exhibits significant potential in enriching neighbor text and enhances accuracy at minimal cost.

II. Related Work

In this section, we present an overview of related research in the fields of GNNs, LLMs as predictors and MQO.

A. Graph Neural Networks

GNNs based on feature-aggregation mechanisms have dominanted graph mining, with representative methods including GCN [35] and GraphSAGE [36]. For node classification tasks

on TAGs, as illustrated in Fig. 1, the conventional GNN-based workflow comprises two steps: first, encoding node text attributes into initial node representations, and second, inputting these representations along with the graph structure into GNNs, which then generate predictions. GNNs are usually trained in a semi-supervised manner. For encoding text attributes, early approaches employed shallow embedding techniques such as Bag-of-Words (BoW) [37] and skip-gram [38]; the current trend favors pre-trained language models (PLMs) that offer richer semantic representations [39]. Additionally, recent research has explored using LLMs to assist GNNs. These approaches leverage LLMs from multiple perspectives, such as providing labeled data [40], enhancing node text attributes [41], and refining graph structures [42], [43]. While GNN-based methods have greatly advanced graph learning, it is important to acknowledge that they encounter numerous challenges in practical scenarios, as previously detailed.

B. LLMs as Predictors

Compared to GNN-based methods, directly querying LLMs to serve as predictors for graph tasks provides significant advantages in terms of scalability and generalization. Existing studies [25]–[27] have determined that common prompt engineering techniques, such as few-shot learning and chain-of-thought (CoT) prompting, do not markedly improve node classification performance. In contrast, incorporating neighbor text into prompts has been demonstrated to enhance accuracy in most datasets. While this can be done directly, some methods [44]–[46] instead attempt to align graph tokens with language tokens via instruction tuning. However, the need for dataset-specific tuning limits their generalization across datasets [45]. Given the high price per input token in black-box LLMs, only a subset of neighbors from the query node's neighborhood can be included in the prompt (see Fig. 1). The primary difference between the methods in this paradigm lies in neighbor selection. For example, some methods randomly select neighbors within a k-hop range [25], while others prioritize neighbors deemed more relevant by LLMs [26] or select neighbors with more similar text attributes [27]. Despite its straightforward nature and still being in the nascent stages of research, this paradigm has already shown notable performance. A recent study [27] indicates that its methods have outperformed traditional GNNs in a comprehensive way, even achieving state-of-the-art (SOTA) results in some datasets. However, the extensive costs of executing multiple node queries via LLMs create a major barrier to their deployment in industrial-scale graphs. Therefore, developing targeted query optimization strategies for this paradigm becomes essential.

C. Multi-Query Optimization (MQO)

MQO is a classical problem in database management systems, where the objective is to minimize the total cost of executing multiple queries by generating a globally optimized query plan [47]. Traditional strategies to MQO have primarily centered on recognizing and reusing shared intermediate results among queries [28]–[30], such as executing a common subexpression only once and reusing the result across multiple queries to reduce redundant computations. With the widespread deployment of LLMs across various applications and considering their substantial inference costs, there is a growing need for MQO strategies tailored for LLM queries [48]. Furthermore, traditional MQO, with its primary focus on SQL query optimization, typically aims solely at reducing costs without emphasizing accuracy. For LLM queries, it is more practical to set the optimization goal to maximize accuracy within a fixed token budget, as LLM predictions are optimizable and the budget is usually fixed. Recent developments [31]–[33] in this field have leveraged the strategy of reusing shared prefixes across multiple queries to minimize the costs of LLM inference. Techniques such as column reordering and row sorting have been enhanced within SQL workloads to maximize prefix sharing efficiency [49]. However, these prefix sharing optimization methods typically require that LLMs be white boxes, which is not feasible in the "LLMs as predictors" paradigm where LLMs are commonly treated as black boxes. Fortunately, the unique structural properties of graph data provide novel opportunities for optimization. This work leverages these properties to develop two MQO strategies that optimize LLM execution of multiple node queries, adaptable to both black-box and white-box models. To the best of our knowledge, this is the first MQO work for LLMs processing graph tasks.

TABLE I
COMMON METHODS AND THEIR NEIGHBOR TEXT. $\mathcal{N}^k(v_i)$ DENOTES NEIGHBORS WITHIN K-HOPS OF v_i, AND TEXT$(\cdot)$ DENOTES THEIR TEXT.

Method	Neighbor Text $\mathcal{N}_i$
vanilla zero-shot	$\varnothing$
1-hop random	Text(RandomSample($\mathcal{N}^1(v_i), M$))
2-hop random	Text(RandomSample($\mathcal{N}^2(v_i), M$))

III. PRELIMINARY

This section outlines the foundational concepts in this work.

A. LLMs as Predictors

A TAG can be defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{X})$, where $\mathcal{V}$, $\mathcal{E}$, $\mathcal{T}$ and $\mathcal{X}$ are node set, edge set, text set and input feature set, respectively. Each node $v_i \in \mathcal{V}$ has a corresponding text attribute $t_i \in \mathcal{T}$ and input feature $x_i \in \mathcal{X}$, where $x_i \in \mathbb{R}^d$ is encoded from t_i through methods like BoW. An edge $e_{ij} = (v_i, v_j) \in \mathcal{E}$ indicates a connection between nodes v_i and v_j. Given some labeled nodes $\mathcal{V}_L \subseteq \mathcal{V}$ with their corresponding labels $\mathcal{Y}_L = \{y_i \mid v_i \in \mathcal{V}_L, y_i \in \{1, \ldots, K\}\}$, where K is the number of classes, the node classification task is to predict the labels of a set of unlabeled nodes $\mathcal{V}_Q \subseteq (\mathcal{V} \setminus \mathcal{V}_L)$.

As shown in Fig. 1, for a single query node $v_i \in \mathcal{V}_Q$, the first step of the "LLMs as predictors" paradigm is neighbor selection. Let M denote the maximum number of neighbors that can be included in the prompt. Different methods select up to M neighboring nodes according to their rules, with the text attributes and labels (if available) of these selected

neighbors constituting the neighbor text $\mathcal{N}_i$. Table I shows the $\mathcal{N}_i$ associated with commonly used methods.

A prompt composing t_i, $\mathcal{N}_i$, and the task description is constructed to query the LLM for the query node's category. The process is formalized as follows:

$$\hat{y}_i = \text{LLM}(t_i, \mathcal{N}_i; prompt), \tag{1}$$

where $\text{LLM}(\cdot; prompt)$ denotes the operation of querying the LLM with the prompt. $\hat{y}_i$ represents the pseudo-label predicted by the LLM's response.

B. Problem Formulation

Here, we formalize the problem of MQO. The optimization process aims to construct an execution plan π for the query set $\mathcal{V}_Q$ that adheres to the overall token budget B while maximizing the overall task performance. The problem can be expressed as follows:

$$\begin{aligned} \pi^* &= \arg\max_{\pi} \sum_{v_i \in \mathcal{V}_Q} \mathbf{1}(y_i = \hat{y}_i), \\ s.t., &\sum_{v_i \in \mathcal{V}_Q} \text{Tokens}(\pi \circ v_i) \leq B, \end{aligned} \tag{2}$$

where $\mathbf{1}(condition)$ denotes the indicator function that equals 1 if the $condition$ is true and 0 otherwise. The function $\text{Tokens}(\pi \circ v_i)$ represents the number of tokens consumed by query node v_i under execution plan π.

IV. Single Query Level Analysis

In this section, we conduct an analysis at the single query level from an information-theoretic perspective, setting the stage for devising optimization strategies for multiple queries.

A. Analyzing Single Query Through Information Theory

Mutual information, denoted as $\text{I}(x; y)$, quantifies the amount of information that the input variable x provides about the label y. For a single query node v_i, the mutual information between its text attribute t_i and its label y_i is denoted as $\text{I}(t_i; y_i)$. With the consumption of additional tokens to include neighbor text $\mathcal{N}_i$, the corresponding mutual information between the combined texts and the label is denoted as $\text{I}(t_i, \mathcal{N}_i; y_i)$.

Definition 1 (Information Gain): For a node v_i, the additional information gained by including $\mathcal{N}_i$ is defined as $\text{IG}^{\mathcal{N}_i} = \text{I}(t_i, \mathcal{N}_i; y_i) - \text{I}(t_i; y_i)$. Given two sets of neighbor text, $\mathcal{N}_i$ and $\tilde{\mathcal{N}}_i$, if $\text{IG}^{\mathcal{N}_i} > \text{IG}^{\tilde{\mathcal{N}}_i}$, it indicates that compared to $\tilde{\mathcal{N}}_i$, $\mathcal{N}_i$ allows a better prediction of y_i.

For an in-depth analysis of $\text{IG}^{\mathcal{N}_i}$, we begin by employing Partial Information Decomposition (PID) [50] on $\text{I}(t_i, \mathcal{N}_i; y_i)$, which specializes in analyzing the multivariate mutual information that a set of source variables contains about a target variable. As shown in Fig. 2, $\text{I}(t_i, \mathcal{N}_i; y_i)$ is decomposed into the following four terms:

$$\begin{aligned} \text{I}(t_i, \mathcal{N}_i; y_i) = &\text{R}(t_i, \mathcal{N}_i; y_i) + \text{U}(t_i \backslash \mathcal{N}_i; y_i) \\ &+ \text{U}(\mathcal{N}_i \backslash t_i; y_i) + \text{S}(t_i, \mathcal{N}_i; y_i). \end{aligned} \tag{3}$$

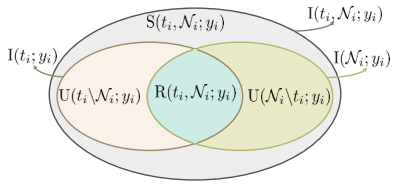

Fig. 2. Partial Information Decomposition of $\text{I}(t_i, \mathcal{N}_i; y_i)$. See Eq. 3 and the following content for detailed explanations.

Here, $\text{R}(t_i, \mathcal{N}_i; y_i)$ denotes redundant information about y_i present in both t_i and $\mathcal{N}_i$. $\text{U}(t_i \backslash \mathcal{N}_i; y_i)$ represents the unique information about y_i found only in t_i and not in $\mathcal{N}_i$, while $\text{U}(\mathcal{N}_i \backslash t_i; y_i)$ represents the unique information about y_i found only in $\mathcal{N}_i$ and not in t_i. $\text{S}(t_i, \mathcal{N}_i; y_i)$ indicates the synergistic information about y_i that emerges only from the combination of $(t_i, \mathcal{N}_i)$ and not from either alone.

Based on the definitions of PID in Eq. 3, $\text{I}(t_i; y_i)$ can be derived as [50]:

$$\text{I}(t_i; y_i) = \text{R}(t_i, \mathcal{N}_i; y_i) + \text{U}(t_i \backslash \mathcal{N}_i; y_i). \tag{4}$$

Thus, the following identity holds exactly:

$$\begin{aligned} \text{IG}^{\mathcal{N}_i} &= \text{I}(t_i, \mathcal{N}_i; y_i) - \text{I}(t_i; y_i) \\ &= \text{U}(\mathcal{N}_i \backslash t_i; y_i) + \text{S}(t_i, \mathcal{N}_i; y_i). \end{aligned} \tag{5}$$

This illustrates that, compared to using only the node's own text t_i, the information gain $\text{IG}^{\mathcal{N}_i}$ of including neighbor text is driven by the unique information in $\mathcal{N}_i$ and the synergistic information between t_i and $\mathcal{N}_i$.

It is important to note that $\text{U}(\mathcal{N}_i \backslash t_i; y_i)$ and $\text{S}(t_i, \mathcal{N}_i; y_i)$ can only add information about y_i if this information is not included in t_i. In other words, although more tokens are consumed to incorporate neighbor text, neighbor text provide unique or synergistic contributions only when t_i itself lacks that specific information about y_i. Therefore, the upper bound of $\text{IG}^{\mathcal{N}_i}$ is given by:

$$\text{IG}^{\mathcal{N}_i} \leq \text{H}(y_i) - \text{I}(t_i; y_i) = \text{H}(y_i | t_i), \tag{6}$$

where the entropy $\text{H}(y_i)$ quantifies the total uncertainty about y_i, and $\text{H}(y_i | t_i)$ denotes the remaining uncertainty about y_i after incorporating the information from t_i.

Definition 2 (Saturated and Non-Saturated Nodes): A node v_i is defined as **saturated**[1] if $\text{H}(y_i | t_i) = 0$ indicating that t_i contains all the necessary information about y_i. Conversely, a node is considered **non-saturated** if $\text{H}(y_i | t_i) > 0$ meaning that t_i does not contain sufficient information about y_i.

For a saturated node, LLMs can make accurate predictions using just it's text t_i, so including neighbor text $\mathcal{N}_i$ merely wastes tokens without providing any information gain. Meanwhile, for a non-saturated node, allocating extra tokens to include neighbor text that can supplement the missing information is a worthwhile investment.

[1]The concept of saturated nodes is applicable to all LLMs, though the specific nodes identified as saturated may differ as the performance of different LLMs may vary.

Principle 1 (Information Gain Principle): In handling a single query node v_i, to avoid unnecessary token consumption and optimize prediction outcomes, the guiding principle for including neighbor text is:

$$\mathcal{N}_i^* = \begin{cases} \varnothing, & \text{if } v_i \text{ is a saturated node,} \\ \arg\max_{\mathcal{N}_i} \mathrm{IG}^{\mathcal{N}_i}, & \text{if } v_i \text{ is a non-saturated node.} \end{cases} \tag{7}$$

B. Optimization Insights

As neighbor text is both a core design component of existing methods and the main token cost, it is key to optimizing both cost and precision. The *principle 1* on neighbor text offers essential optimization guidance, which we outline here.

1) Omitting Neighbor Text for Saturated Nodes: Existing methods add neighbor text into the prompt uniformly for all queries, without distinguishing between saturated and non-saturated nodes. However, as *principle 1* indicates, for any saturated node v_i, adding neighbor text $\mathcal{N}_i$ merely wastes tokens. Moreover, prior work found that *k-hop random* can underperform *vanilla zero-shot* on datasets like Pubmed [25], [27]. Given the high proportion of saturated nodes, as evidenced by the 90% accuracy of *vanilla zero-shot* on Pubmed, we infer that incorporating neighbor text for saturated nodes is not only redundant but may also introduce noise.

In previous studies, *vanilla zero-shot* ($\mathcal{N}_i = \varnothing$) demonstrated decent performance, even with the entry-level LLM, LLaMA-2-7B, achieving about 50% accuracy across various datasets [26]. This suggests that saturated nodes, which do not need neighbor text, are widely present in the datasets, even when using the "simplest" LLM. Existing methods expend considerable tokens on these nodes unnecessarily.

When facing budget constraints, identifying saturated nodes among multiple node queries and omitting their neighbor text can effectively reduce token costs of existing methods without degrading task performance, as described in Section V-A.

2) Increasing Neighbor Information Gain for Non-Saturated Nodes: For each non-saturated node v_i, the optimization objective should focus on enriching its neighbor text to increase $\mathrm{IG}^{\mathcal{N}_i}$, and this should be achieved with minimal token usage. A practical and cost-effective solution is to enrich the neighbor text with neighbor labels; these brief labels consume minimal tokens and generally benefit predictions. This strategy is supported by the widely recognized homophily principle [34], which suggests that neighboring nodes are likely to share the same label y_i. Consequently, including neighbor labels is likely to increase $\mathrm{U}(\mathcal{N}_i \backslash t_i; y_i)$ and $\mathrm{S}(t_i, \mathcal{N}_i; y_i)$, thereby enhancing $\mathrm{IG}^{\mathcal{N}_i}$.

An exploratory experiment was conducted to assess the effectiveness and optimization potential of this solution, specifically questioning whether neighbor labels benefit predictions and if existing methods' queries frequently lack neighbor labels. Each query was executed by two kinds of methods: *k-hop random* and *vanilla zero-shot*. *k-hop random* equips query with neighbor text, while *vanilla zero-shot* does not. Thus, they each correspond to one of the terms in $\mathrm{IG}^{\mathcal{N}_i} = \mathrm{I}(t_i, \mathcal{N}_i; y_i) - \mathrm{I}(t_i; y_i)$. Accordingly, the accuracy gain of *k-hop random* over *vanilla zero-shot* can be used as a proxy for $\mathrm{IG}^{\mathcal{N}_i}$, which quantifies the information gain from including neighbor text. Let $\mathcal{N}_i^L = \mathcal{N}_i \cap \mathcal{V}_L$ denote the set of labeled nodes within $\mathcal{N}_i$. $\mathcal{N}_i^L = \varnothing$ indicates that the neighbor text does not contain neighbor labels, and vice versa for $\mathcal{N}_i^L \neq \varnothing$. (1) As shown in the bar charts in Fig. 3, queries with $\mathcal{N}_i^L \neq \varnothing$ consistently exhibit higher $\mathrm{IG}^{\mathcal{N}_i}$ compared to queries with $\mathcal{N}_i^L = \varnothing$, confirming the effectiveness of neighbor labels in increasing $\mathrm{IG}^{\mathcal{N}_i}$. (2) As depicted in the pie charts in Fig. 3, queries with $\mathcal{N}_i^L = \varnothing$ are numerous, demonstrating that existing methods often lack neighbor labels in their queries.

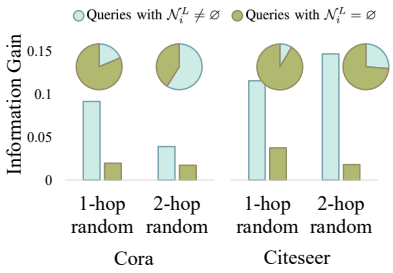

Fig. 3. The bar charts depict the impact of neighbor labels in increasing $\mathrm{IG}^{\mathcal{N}_i}$ in the Cora and Citeseer datasets. Each pie chart shows the proportion of queries with $\mathcal{N}_i^L \neq \varnothing$ and $\mathcal{N}_i^L = \varnothing$.

This implies that enriching neighbor text with neighbor labels could enhance task performance without incurring substantial token costs, as illustrated in Section V-B.

V. Proposed MQO Strategies

This section presents two MQO strategies: token pruning and query boosting, which are derived from the optimization insights previously described.

A. Token Pruning Strategy

Since omitting neighbor text for saturated nodes can reduce token consumption, it is essential to identify these nodes from all query nodes. Identifying saturated nodes requires evaluating whether an LLM can correctly classify a node based solely on its text attribute, usually assessed via prediction uncertainty. A direct solution is to query the LLM using just the text attribute ($\mathrm{LLM}(t_i; \text{prompt})$), and the log probability that the LLM assigns to the category token can serve to quantify uncertainty. However, this solution incurs extra query costs, contradicting our goal of cost-saving.

1) Text Inadequacy Measure: Our method instead breaks down the LLM's uncertainty into two components: one stemming from the ambiguity of the node's text attributes across categories, and the other from the LLM's inherent bias in categories. Compared to non-saturated nodes, saturated nodes have text attributes with clearer category information, and their categories align with those that the LLM excels at predicting. We first calculate the two components separately and then integrate them into an overall text inadequacy measure, $\mathrm{D}(t_i)$, utilized for identifying saturated nodes.

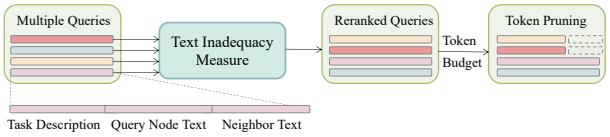

Fig. 4. Schematic diagram of the token pruning strategy.

Using the entropy of classification probabilities is a common method for assessing the ambiguity of input. However, obtaining these probabilities from black-box LLMs is impractical and costly. To simplify the approach, we use a MLP as a surrogate classifier f_{θ_1}, inputting node features x_i which are encoded from t_i via text encoding methods. The class probabilities output by f_{θ_1} are then used to compute entropy. The surrogate classifier f_{θ_1} is trained on the labeled set $\mathcal{V}_L$, by solving: ${\theta_1}^* = \arg\min_{\theta_1} \sum_{v_i \in \mathcal{V}_L} \ell(f_{\theta_1}(x_i), y_i)$, where $\ell(\cdot)$ represents the cross-entropy loss function. Then, for each query node v_i, we can compute the entropy of its category probability distribution $\boldsymbol{p_i}$, given by:

$$\mathrm{H}(\boldsymbol{p_i}) = \mathrm{H}(f_{\theta_1^*}(x_i)) = \mathrm{H}(p_{i,1}, \ldots, p_{i,K}). \tag{8}$$

To assess bias in LLMs toward certain classes, we first randomly select a small subset of nodes, $\mathcal{V}_L^c$, from $\mathcal{V}_L$ and use their text attributes to generate LLM predictions, represented by $\hat{y}_i = \mathrm{LLM}(t_i; \text{prompt})$. Based on these predictions on $\mathcal{V}_L^c$, we calculate the distribution of misclassification ratios for each class, denoted as $\boldsymbol{w} = (w_1, \ldots, w_K)$, where each w_k is computed as: $w_k = \frac{\sum_{v_i \in \mathcal{V}_L^c} \mathbf{1}(y_i=k) \cdot \mathbf{1}(\hat{y}_i \neq y_i)}{\sum_{v_i \in \mathcal{V}_L^c} \mathbf{1}(y_i=k)}$. Then, for each query node v_i, we can compute the inadequacy arising from LLMs' category bias, given by:

$$b_i = \boldsymbol{p_i} \times \boldsymbol{w}^{\mathrm{T}}. \tag{9}$$

Finally, we train a linear regression model, denoted as g_{θ_2}, to merge the two inadequacy channels into a comprehensive measure of inadequacy. This is accomplished by solving: ${\theta_2}^* = \arg\min_{\theta_2} \sum_{v_i \in \mathcal{V}_L^c} \left(\mathbf{1}(\hat{y}_i \neq y_i) - g_{\theta_2}(\mathrm{H}(\boldsymbol{p_i}) \parallel b_i)\right)^2$, where $\parallel$ denotes concatenation. Thus, the overall text inadequacy $\mathrm{D}(t_i)$ of v_i is given by:

$$\mathrm{D}(t_i) = g_{\theta_2^*}(\mathrm{H}(\boldsymbol{p_i}) \parallel b_i). \tag{10}$$

As a proxy for $\mathrm{H}(y_i|t_i)$, $\mathrm{D}(t_i)$ helps identify saturated vs. non-saturated nodes among multiple query nodes, with smaller values for saturated and larger values for non-saturated.

2) Token Pruning: As depicted in Fig. 4, when faced with a limited token budget, we first calculate the text inadequacy $\mathrm{D}(t_i)$ for each query node $v_i \in \mathcal{V}_Q$ according to Eq. 10 and order them in ascending order. The saturated nodes are expected to be ranked at the front, allowing for the pruning of their neighbor text to save tokens. Meanwhile, the remaining nodes are considered non-saturated and are more likely to benefit from the neighbor text.

Users can adjust the number of node queries pruning neighbor text according to their token budgets; put simply, the budget B determines the percentage ($\tau\%$) of top-ranked queries to omit neighbor text. For scenarios with a limited token budget, *Algorithm 1* details the token pruning strategy.

Algorithm 1 Token Pruning Strategy

Input: Query set $\mathcal{V}_Q$; token budegt B; LLM; surrogate classifier $f_{\theta_1^*}$; linear regression model $g_{\theta_2^*}$.

```
# Step 1: Calculate Text Inadequacy
for each query node v_i ∈ V_Q do
    Compute H(p_i) according to Eq. 8.
    Compute b_i according to Eq. 9.
    Compute D(t_i) according to Eq. 10.
end for
# Step 2: Rank and Token Pruning
Rank nodes in V_Q based on D(t_i) in ascending order.
Determine the percentage (τ%) of queries to omit neighbor text
based on budget B.
for each query in the top τ% of queries do
    ŷ_i = LLM(t_i; prompt).
end for
for each query outside the top τ% of queries do
    ŷ_i = LLM(t_i, N_i; prompt).
end for
```

Output: Predicted labels $\hat{\mathcal{Y}}_Q = \{\hat{y}_i \mid v_i \in \mathcal{V}_Q\}$.

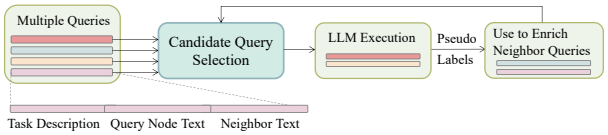

Fig. 5. Schematic diagram of the query boosting strategy.

B. Query Boosting Strategy

Acknowledging the potential to boost task performance by incorporating neighbor labels into neighbor text, we propose using pseudo-labels from earlier queries to enrich the neighbor text of subsequent neighboring queries, leveraging the interconnected nature in multiple node queries. Once earlier queries are executed, their nodes and generated pseudo-labels expand the labeled set $\mathcal{V}_L$ and the labels set $\mathcal{Y}_L$. Subsequently, the neighbor text of unexecuted queries is updated, incorporating these pseudo-labels based on the latest $\mathcal{V}_L$ and $\mathcal{Y}_L$.

In pursuit of optimal pseudo-label utilization, our strategy includes a query scheduling algorithm designed with dual goals: preventing the spread of incorrect pseudo-labels and increasing the number of enriched queries. It organizes queries into sequential rounds, placing those with multiple reliable neighbor labels in earlier rounds because these are likely to receive accurate pseudo-labels. Conversely, queries with sparse or conflicting neighbor labels are placed in later rounds, which not only increases their chance of enrichment by pseudo-labels from prior rounds but also minimize their pseudo-label propagation, considering their greater risk of inaccuracies.

Specifically, in each round, we select queries from the unexecuted pool to constitute the candidate query set $\mathcal{C}$, considering both the number of neighbor labels and any conflicts between them. The candidate criterion is defined as follows:

$$\mathcal{C} = \{v_i \mid |\mathcal{N}_i^L| \geq \gamma_1 \text{ and } LC_i \leq \gamma_2\}, \tag{11}$$

Algorithm 2 Query Boosting Strategy

Input: Set $\mathcal{V}_L$ with labels $\mathcal{Y}_L$; query set $\mathcal{V}_Q$; γ_1, γ_2.

while $\mathcal{V}_Q \neq \varnothing$ **do**
  # Step 1: Candidate Query Selection
  Initialize candidate query set $\mathcal{C} \leftarrow \varnothing$
  repeat
    for each $v_i \in \mathcal{V}_Q$ **do**
      # Enrich Neighbor Text with Pseudo-Labels
      Refresh $\mathcal{N}_i$ with the latest $\mathcal{V}_L$ and $\mathcal{Y}_L$.
      Count neighbor labels $|\mathcal{N}_i^L|$.
      Count conflicting labels LC_i.
      if $|\mathcal{N}_i^L| \geq \gamma_1$ **and** $LC_i \leq \gamma_2$ **then**
        Add v_i to candidate query set $\mathcal{C}$.
      end if
    end for
    if $\mathcal{C} = \varnothing$ **then**
      $\gamma_1 \leftarrow \gamma_1 - 1$ or $\gamma_2 \leftarrow \gamma_2 + 1$
    else
      Exit the repeat loop
    end if
  until $\mathcal{C} \neq \varnothing$
  # Step 2: Query LLM
  for each $v_i \in \mathcal{C}$ **do**
    $\hat{y}_i = \mathrm{LLM}(t_i, \mathcal{N}_i; \mathrm{prompt})$
  end for
  # Step 3: Add Pseudo-Labels to Labeled Set
  for each $v_i \in \mathcal{C}$ **do**
    Add v_i to $\mathcal{V}_L$, add $\hat{y}_i$ to $\mathcal{Y}_L$
    Remove v_i from $\mathcal{V}_Q$
  end for
end while

Output: Predicted labels $\hat{\mathcal{Y}}_Q = \{\hat{y}_i \mid v_i \in \mathcal{V}_Q\}$.

where $|\mathcal{N}_i^L|$ indicates the number of neighbor labels, $LC_i = |\{y_j \mid v_j \in \mathcal{N}_i^L\}|$ counts the different types of conflicting labels. The integer hyperparameters, γ_1 and γ_2, which denote the thresholds for neighbor labels and conflicting labels respectively. During later stages of this process, there may be rounds where none of unexecuted queries meet the candidate criteria ($|\mathcal{N}_i^L| \geq \gamma_1$ and $LC_i \leq \gamma_2$), leading to an empty candidate query set $\mathcal{C} = \varnothing$. In such cases, we relax the values of γ_1 and γ_2 incrementally (e.g., $\gamma_1 \leftarrow \gamma_1 - 1$ or $\gamma_2 \leftarrow \gamma_2 + 1$). This enables us to broaden the selection criteria while still prioritizing those queries that are most likely to be correctly predicted. The incremental nature of this relaxation ensures that we continue to leverage the core benefit of our query boosting strategy—using reliable pseudo-labels to refine the prediction process in subsequent rounds.

A simplified schematic of this query boosting strategy is depicted in Fig. 5. *Algorithm 2* outlines the entire procedure.

C. A Running Example

In this section, we present a running example to illustrate the two optimization strategies. As shown in Fig. 6, the query set $\mathcal{V}_Q$ consists of papers to be sent to the LLM for category prediction. Each query contains the title and abstract of the paper, along with the titles and abstracts of its referenced neighbor papers and a problem description. The average number of tokens consumed per query is denoted as

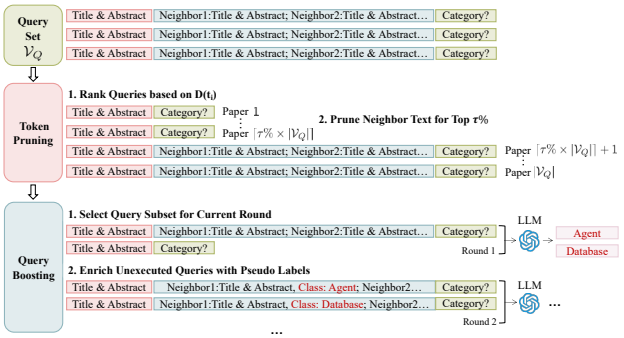

Fig. 6. A running example of our optimization strategies, which can be applied in sequence or individually.

$\overline{\mathrm{Tokens}(v)}$, with the average number of tokens consumed by the neighbor text represented by $\overline{\mathrm{Tokens}(\mathcal{N})}$[2].

1) Token Pruning: Assume the user's limited token budget $B < \overline{\mathrm{Tokens}(v)} \times |\mathcal{V}_Q|$, meaning some queries must omit their neighbor text to save tokens. The proportion of such queries, $\tau\%$, is determined as follows: $B = \tau\% \times |\mathcal{V}_Q| \times \left(\overline{\mathrm{Tokens}(v)} - \overline{\mathrm{Tokens}(\mathcal{N})}\right) + (1 - \tau\%) \times |\mathcal{V}_Q| \times \overline{\mathrm{Tokens}(v)}$, solving for $\tau\%$ gives: $\tau\% = \frac{|\mathcal{V}_Q| \times \overline{\mathrm{Tokens}(v)} - B}{|\mathcal{V}_Q| \times (\overline{\mathrm{Tokens}(v)} - \overline{\mathrm{Tokens}(\mathcal{N})})}$.

For each query paper $v_i \in \mathcal{V}_Q$, its text inadequacy score $\mathrm{D}(t_i)$ is calculated according to Eq. 10. The queries are then ranked based on $\mathrm{D}(t_i)$. For the top $\tau\%$ of queries, as shown in Fig. 6, the neighbor text is omitted.

2) Query Boosting: In each round, queries are selected from the unexecuted query pool via Eq. 11 and sent to the LLM for category prediction. The resulting pseudo-labels (e.g., "Agent", "Database") then enrich the neighbor text of subsequent neighboring queries, as shown in Round 2 of Fig. 6, where the neighbor text includes the newly added "class: Agent". This continues until all queries are classified.

3) Plug-and-Play Integration: Both token pruning and query boosting serve as plug-and-play components that can be seamlessly integrated into existing "LLMs as predictors" methods. Specifically, token pruning optimizes token usage by omitting neighbor text from the prompts of saturated nodes, while query boosting improves query informativeness by appending pseudo-labels to the prompt. Notably, both strategies operate on query prompts rather than modifying the model itself, ensuring broad applicability across different frameworks. Moreover, these two strategies can be deployed separately or combined sequentially, offering flexible optimization options tailored to different objectives.

VI. Experiments

This section evaluates the two optimization strategies regarding their effectiveness, universality, and application potential. We start with an overview of the experimental settings before addressing the following specific research questions: **Q1**: Is the token pruning strategy effective across different

[2]Both values can be estimated through statistical analysis or approximation.

methods? **Q2**: Does the token pruning strategy adapt well to different budgets? **Q3**: How many tokens can be reduced by the token pruning strategy in ideal conditions? **Q4**: How effective is the designed text-inadequacy measure in identifying saturated nodes? **Q5**: Is the query scheduling algorithm capable of enhancing the utilization of pseudo-labels? **Q6**: Is the query boosting strategy effective across various methods? **Q7**: How effective is the joint application of the two strategies? **Q8**: Can the two strategies be applied broadly across various frameworks? **Q9**: How effective are the two strategies in the link prediction task?

A. Experimental Settings

1) Datasets: We conduct evaluations on the following five commonly used TAG datasets: Cora [51], Citeseer [52], Pubmed [53], Ogbn-Arxiv [54], and Ogbn-Products [54]. Table II presents the statistical information for each dataset. Node classification categorizes papers and products, while link prediction predicts citation and co-purchase relationships. These fundamental graph tasks play a crucial role in various applications, such as recommendations and reasoning. Our dataset processing and partitioning adhere to those used in prior research within the "LLMs as predictors" paradigm [25]–[27]. For the Cora, Citeseer, and Pubmed datasets, we use 20 labeled nodes per category as $\mathcal{V}_L$ and randomly select 1,000 unlabeled nodes to form the query set $\mathcal{V}_Q$, which is used for querying the LLM regarding their categories. For the Ogbn-Arxiv and Ogbn-Products datasets, we follow the original dataset partitions but also randomly select 1,000 nodes from the test set to form the query set $\mathcal{V}_Q$.

2) Benchmark Methods for Optimization: Our optimization strategies are assessed using prevalent "LLMs as predictors" methods. These benchmark methods primarily differ in neighbor selection, while all other steps remain consistent. The neighbor selection details for each method are as follows:

- *Vanilla zero-shot*: No neighbor text is incorporated.
- *k-hop random, k=1,2*: Neighbors are selected within the k-hop range of the query node, with a preference for labeled neighbors followed by a random selection from unlabeled neighbors, up to a fixed number limit, M.
- *SNS* [27]: This method progressively explores from closer to farther hops to find enough labeled neighbors or until reaching five hops. It then uses SimCSE [55] to measure and rank the similarity between the query node's text and the identified labeled neighbors. The top-ranking neighbors are selected in order, up to a limit of M.

Table III exemplifies the prompt templates. To save tokens, the default configuration for neighbor text includes only the titles and labels (if available) of the selected neighbors.

3) Implementations: Here, we present the implementation details of benchmark methods and our optimization strategies.

- Regarding benchmark methods, GPT-3.5-0125 (as the default) and GPT-4.0-mini serve as LLMs for executing queries. The maximum number of neighbors per prompt, M, is set to 10 for the Ogbn-Products dataset and 4 for all other datasets, although some nodes may include fewer neighbors due to having a lower degree.
- Regarding optimization strategies, since our experimental design emphasizes validating the versatility of our optimization strategies, we avoid hyperparameter tuning as much as possible. In the query boosting strategy, we set the neighbor labels threshold γ_1 and the conflicting labels threshold γ_2 at 3 and 2, respectively, for all datasets and benchmark methods. In the token pruning strategy, for small-scale datasets (Cora, Citeseer, Pubmed), we use a linear MLP as the surrogate classifier f_{θ_1}, with a consistent learning rate of 0.01 and no weight decay. For large-scale datasets (Ogbn-Arxiv, Ogbn-Products), where sufficient labeled data is available, we conduct a simple hyperparameter search for f_{θ_1} within the following ranges: number of layers $\{2, 3\}$, hidden size $\{96, 128, 256\}$, learning rate $\{0.001, 0.01\}$, and weight decay $\{0.0001, 0.001\}$. We employ 3-fold cross-validation to obtain the average category probability distribution and entropy. The subset $\mathcal{V}_L^c$, employed to evaluate the category bias of LLMs, is configured to be 10 times the number of classes.

B. Effectiveness of the Token Pruning Strategy Across Different Benchmark Methods (*Q1*)

In each dataset, we utilize three benchmark methods to execute 1,000 node classification queries as a baseline comparison. In parallel, we implement our token pruning strategy on these methods. This process begins by sorting these queries in ascending order by their text-inadequacy scores $\mathrm{D}(t_i)$, targeting saturated nodes that do not rely on neighbor text for high ranking. Then, we omit neighbor text from the top 20% of these sorted queries for each method. Finally, each method executes the queries, with the top 20% lacking neighbor text and the remainder keeping it.

Table IV presents the classification results, where "w/ token prune" denotes the versions that apply the token pruning strategy, reducing token costs by omitting neighbor text in 20% of the queries. $\Delta\%$ indicates the percentage change in accuracy of the "w/ token prune" version relative to the original method. Across various datasets and methods, the accuracy of the "w/ token prune" versions does not exhibit significant degradation compared to their original counterparts, as evidenced by the negligible $\Delta\%$. This suggests that the top 20% of queries selected based on our designed text-inadequacy measure indeed do not need neighbor text. Furthermore, this confirms the viability of our token pruning strategy as both plug-and-play and universally applicable, effectively optimizing token usage across various methods without diminishing task performance.

C. Effectiveness of the Token Pruning Strategy Under Different Token Budgets (*Q2*)

For each dataset, we set a series of token budgets for 1,000 node classification queries, allowing neighbor text inclusion in up to 0%, 20%, 40%, 60%, 80%, 100% of queries. This necessitates selecting a corresponding proportion of queries

TABLE II
STATISTICS OF DATASETS.

Dataset	#Nodes	#Edges	#Features	#Classes	Node Type	Text Type	Edge Type	Category Labels
Cora	2,708	5,429	1,433	7	Paper	Title&Abstract	Citation	[Case-Based, Theory, ...]
Citeseer	3,186	4,277	500	6	Paper	Title&Abstract	Citation	[Database, Agents, ...]
Pubmed	19,717	44,338	384	3	Paper	Title&Abstract	Citation	[Type 1 diabetes, Type 2...]
Ogbn-Arxiv	169,343	1,166,243	128	40	Paper	Title&Abstract	Citation	[cs.AI, cs.CL, ...]
Ogbn-Products	2,449,029	61,859,140	100	47	Product	Description	Co-purchase	[Books, Beauty, ...]

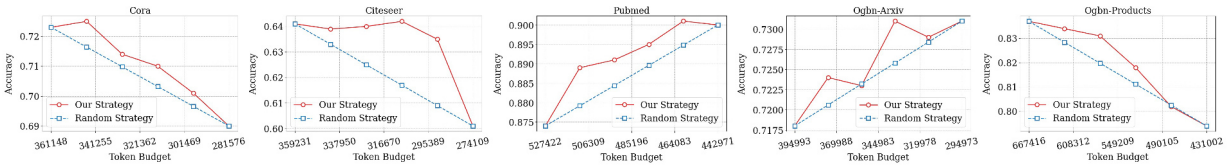

Fig. 7. Classification accuracy comparison between our token pruning strategy and the random strategy on the 1-hop random method, across token budgets that allow neighbor text inclusion for up to 100%, 80%, 60%, 40%, 20%, 0% of queries.

TABLE III
PROMPT TEMPLATES USED FOR QUERYING LLMS.

vanilla zero-shot
Target paper: Title: `title` \nAbstract: `abstract` \nTask: \nCategories: \n[`category 1`,...,`category K`]\nWhich category does the target paper belong to?\nPlease output the most likely category as a Python list: Category: ['XX'].
1-hop random, 2-hop random, SNS
Target paper: Title: `title` \nAbstract: `abstract` \n\nTarget paper has the following important neighbors with citation relationships (if SNS, add "from most related to least related"):\nNeighbor Paper0: {{\nTitle: `paper 0 title` \nCategory: `paper 0 label` \n}}\nNeighbor Paper1: {{\nTitle: `paper 1 title` \n}}\n... (more k-hop neighbors) \nTask: \nCategories: \n[`category 1`,...,`category K`]\nWhich category does the target paper belong to?\nPlease output the most likely category as a Python list: Category: ['XX'].

TABLE IV
CHANGES IN CLASSIFICATION ACCURACY(%) RESULTING FROM TOKEN REDUCTION USING THE TOKEN PRUNING STRATEGY.

	Cora	Citeseer	Pubmed	Ogbn-Arxiv	Ogbn-Products
1-hop random	72.3	64.1	87.4	71.8	83.7
w/ token prune	72.5	63.9	88.9	72.4	83.4
Δ%	0.28%	-0.31%	1.72%	0.84%	-0.36%
2-hop random	72.0	64.8	88.8	72.6	83.5
w/ token prune	71.9	64.5	89.1	72.9	83.0
Δ%	-0.14%	-0.46%	0.34%	0.41%	-0.60%
SNS	74.8	69.3	89.3	71.5	84.3
w/ token prune	74.4	68.5	88.8	71.8	84.0
Δ%	-0.53%	-1.15%	-0.56%	0.42%	-0.36%

from which neighbor text must be omitted. The first strategy randomly selects the requisite number of queries to omit neighbor text to meet budget constraints. In contrast, our token pruning strategy arranges the queries in ascending order by text-inadequacy scores and selects the requisite portion from the top to omit neighbor text.

Fig. 7 illustrates the classification accuracy of the two strategies across different budget constraints. In scenarios requiring the omission of neighbor text from queries, our token pruning strategy (the red line) consistently results in higher accuracy compared to the random strategy (the blue line). This indicates that our strategy effectively optimizes token allocation when faced with constrained budgets, significantly mitigating the potential negative impacts on prediction outcomes due to budget reductions.

Additionally, we noted that for the Pubmed and Ogbn-Arxiv datasets, scenarios where all queries omitted neighbor text (right endpoint) achieved higher accuracy compared to cases where all queries included neighbor text (left endpoint). This indicates that beyond consuming additional tokens, neighbor text might also introduce noise that impairs the LLM's task performance. Nonetheless, our token pruning strategy still offered a comparative advantage by assisting saturated nodes in avoiding the noise from neighbor text. This benefit is also evident in Table IV, showing that, in most cases on the Pubmed and Ogbn-Arxiv datasets, the accuracy of the "w/ token prune" versions is higher than that of their original counterparts.

D. Token Reduction Potential of Token Pruning (*Q3*)

We calculate the number of tokens that can theoretically be saved, determined by the total number of queries ($|\mathcal{V}|$), the proportion of saturated nodes ($\tau\%$), and the average token consumption of the neighbor text ($\overline{Tokens(\mathcal{N})}$). To estimate the proportion of saturated nodes, we use the *vanilla zero-shot* method to execute 1,000 node classification queries in each dataset, none of which include neighbor text. Its prediction accuracy can serve as a proxy for the proportion of saturated nodes ($\tau\%$), which are nodes whose text attributes alone are adequate for LLMs to make correct predictions. We compute $\overline{Tokens(\mathcal{N})}$ under four types of neighbor text configurations: titles of 4 neighbors; titles of 10 neighbors; titles and abstracts of 4 neighbors; titles and abstracts of 10 neighbors. The product $|\mathcal{V}| \times \tau\% \times \overline{Tokens(\mathcal{N})}$ gives the amount of tokens that can theoretically be reduced.

TABLE V
NUMBER OF TOKENS POTENTIALLY REDUCIBLE VIA TOKEN PRUNING IN VARIOUS NEIGHBOR TEXT CONFIGURATIONS.

		Cora	Citeseer	Pubmed	Ogbn-Arxiv	Ogbn-Products
Accuracy	# Total queries Proportion of saturated nodes	2708 69.0%	3186 60.1%	19,717 90.0%	169,343 73.1%	2,449,029 79.4%
4 Neighbors, Title Only	# Neighbor Text Tokens # Potentially Reducible Tokens	67.718 126,533	122.667 234,880	95.472 1,694,184	66.431 8,223,473	61.745 120,064,767
10 Neighbors, Title Only	# Neighbor Text Tokens # Potentially Reducible Tokens	169.296 316,333	306.667 587,201	238.681 4,235,460	166.077 20,558,683	154.362 300,161,919
4 Neighbors, Title & Abstract	# Neighbor Text Tokens # Potentially Reducible Tokens	799.909 1,494,647	804.518 1,540,480	1291.507 22,918,181	924.565 114,451,613	553.777 1,076,835,422
10 Neighbors, Title & Abstract	# Neighbor Text Tokens # Potentially Reducible Tokens	1999.774 3,736,617	2011.295 3,851,200	3228.768 57,295,452	2311.412 286,129,034	1384.442 2,692,088,554

TABLE VI
AVERAGE TEXT-INADEQUACY VALUES IN SATURATED AND NON-SATURATED NODES.

Node Type	Cora	Citeseer	Pubmed	O.-Arv.	O.-Prdt.
Saturated	0.421	0.350	0.265	0.298	0.144
Non-saturated	0.478	0.437	0.330	0.339	0.253

Statistical results are presented in Table V. Firstly, the *vanilla zero-shot* method performs consistently well across all datasets, achieving accuracy ranging from 60% to 90%. This indicates that in all datasets, many nodes have text attributes that provide enough information for the LLM to make accurate predictions without needing additional neighbor text, suggesting a high proportion of saturated nodes. Secondly, the calculated number of potentially reducible tokens is substantial, particularly evident in large datasets like the Ogbn-Products dataset, where the reducible token count even reaches the order of 2×10^9. This highlights the significant application potential of our token pruning strategy, which advocates omitting neighbor text from saturated nodes. Furthermore, we notice that as the number and content of included neighbors increase, the growth in potentially reducible tokens becomes even more pronounced. This suggests that the token pruning strategy is well-suited for tasks demanding extensive node contexts, such as those involving long-range dependencies.

*E. Effectiveness of the Text-Inadequacy Measure (**Q4**)*

For 1,000 queries in each dataset, we classify them as saturated or non-saturated nodes depending on whether the *vanilla zero-shot* method accurately predicts their categories. We subsequently computed the average values of our text-inadequacy measure, $D(t_i)$, for each group.

As shown in Table VI, the average text-inadequacy values for saturated nodes are consistently lower than those for non-saturated nodes across all datasets. This indicates that our designed text-inadequacy $\mathrm{D}(t_i)$ serves as a reliable proxy for $\mathrm{H}(y_i|t_i)$, with lower values indicating a higher likelihood of node saturation. Additionally, although Table IV and Fig. 7 demonstrate broad optimization advantages from the existing text-inadequacy measure, the disparity in average text-inadequacy values across the two groups remains modest. We attribute this to the challenging nature of the task and the simple heuristic currently used for calculation. Enhancing this measure in future research, possibly by leveraging classification probabilities from LLMs or utilizing more annotated saturated nodes, could be highly valuable, given the considerable potential for token savings shown in Table V.

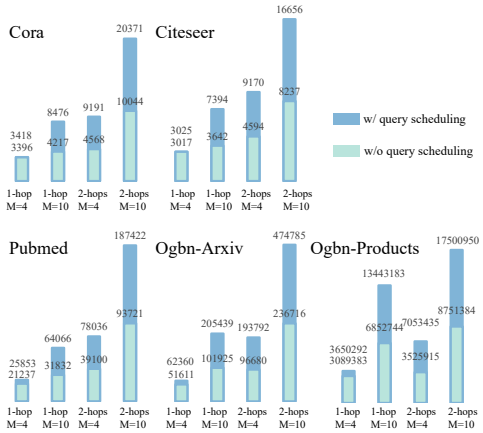

Fig. 8. Comparison of pseudo-label utilization frequencies with and without the use of the query scheduling algorithm.

*F. Effectiveness of the Query Scheduling Algorithm in Enhancing Pseudo-label Utilization (**Q5**)*

For each dataset, we set four configurations of neighbor text, varying the top distance (1-hop or 2-hop) and the maximum number of neighbors (M = 4 or 10). Under each configuration, we compare the pseudo-label utilization between the "w/ query scheduling" and "w/o query scheduling" versions, counting how many times pseudo-labels generated by earlier queries are used to enrich the neighbor text of later queries. The "w/o query scheduling" version randomly divided queries into 50 rounds for execution. For a fair comparison, we also configured "w/ query scheduling" version into 50 rounds. The two versions are largely consistent, except in the execution

TABLE VII
COMPARISON OF CLASSIFICATION ACCURACY (%) BEFORE AND AFTER OPTIMIZATION OF THE QUERY BOOSTING STRATEGY. ↑ DENOTES IMPROVEMENT.

	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
	GPT 4o-mini			*GPT 3.5*		
1-hop random	67.8	61.0	79.4	72.3	64.1	87.4
w/ query boost	68.0↑	63.2↑	79.2	72.8↑	65.3↑	87.9↑
2-hop random	68.9	64.2	80.0	72.0	64.8	88.8
w/ query boost	70.2↑	68.8↑	80.6↑	74.2↑	67.3↑	89.4↑
SNS	72.1	68.9	80.8	74.8	69.3	89.3
w/ query boost	72.6↑	70.6↑	81.6↑	76.3↑	70.6↑	90.3↑

order of queries: while the "w/o query scheduling" version randomly orders queries, "w/ query scheduling" version arranges each round's queries by the count of neighbor labels included, prioritizing those with more neighbor labels among all unexecuted queries[3]. Note that this experiment does not concern classification accuracy; we merely simulate LLMs to generate pseudo-labels for queries rather than incurring substantial costs by actual implementation.

Fig. 8 illustrates pseudo-label utilization for both versions. (1) Even without the query scheduling algorithm, the "w/o query scheduling" version achieves notable pseudo-label utilization across different configurations and datasets, confirming the general feasibility of using pseudo-labels to enrich queries. (2) Configurations with more neighbors and a larger hop range, such as 2-hops and $M = 10$, report higher pseudo-label utilization, as these settings lead to more frequent query associations, offering more enrichment opportunities. (3) In the 1-hop, $M = 4$ configuration, the "w/ query scheduling" version offers only a modest improvement over the "w/o query scheduling" version, due to the inherently fewer associations among queries in this particular setting, which limits the potential for optimization through query scheduling. (4) Across all other configurations and datasets, the "w/ query scheduling" version nearly doubles the pseudo-label utilization compared to the "w/o query scheduling" version, demonstrating the effectiveness of the query scheduling algorithm.

G. Effectiveness of the Query Boosting Strategy Across Different Benchmark Methods (*Q6*)

Query boosting relies on the connections among multiple queries, making it unsuitable when such connections are sparse. For example, in Ogbn-Products, with 1,000 randomly selected queries from 2 million nodes, sparse interconnections provide few opportunities to acquire pseudo-labels from neighboring queries. Moreover, cost constraints prevent us from executing large-scale queries adapted to big datasets to create a suitable experimental setup for query boosting. Therefore, classification tests for the query boosting strategy are limited to smaller datasets: Cora, Citeseer, and Pubmed, using 1,000 queries each. To minimize costs, we set $M = 4$, which, as previously discussed, limits the advantages of query boosting.

[3]While the full candidate criteria include a conflicting labels threshold, it has been omitted here due to the use of simulated pseudo-labels.

TABLE VIII
COMPARISON OF CLASSIFICATION ACCURACY (%) AND TOKEN COST BEFORE AND AFTER APPLYING THE JOINT STRATEGY.

	# Queries Equip $\mathcal{N}_i$	Cora	Citeseer	Pubmed
		GPT 4o-mini		
1-hop random	1000	67.8	61.0	79.4
w/ prune & boost	800	68.5↑	61.9↑	79.3
2-hop random	1000	68.9	64.2	80.0
w/ prune & boost	800	70.3↑	67.9↑	80.8↑
SNS	1000	72.1	68.9	80.8
w/ prune & boost	800	71.9	69.2↑	80.9↑
		GPT 3.5		
1-hop random	1000	72.3	64.1	87.4
w/ prune & boost	800	71.6	65.6↑	89.4↑
2-hop random	1000	72.0	64.8	88.8
w/ prune & boost	800	73.4↑	66.5↑	89.5↑
SNS	1000	74.8	69.3	89.3
w/ prune & boost	800	75.6↑	69.8↑	89.9↑

Table VII reports the classification results. The "w/ query boost" versions employ the query boosting strategy, which utilizes pseudo-labels from earlier queries to enrich later queries through the query scheduling algorithm. Since pseudo-labels are short category names like "Database", adding them to query prompts incurs negligible extra token costs. It can be observed that the query boosting strategy delivers classification gains for all methods, although the limited associations among queries caused by the $M = 4$ configuration restrict its advantages. This not only validates our earlier analysis that neighbor labels benefit prediction but also highlights the query boosting strategy's contribution to optimizing classification accuracy for various methods.

H. Effect of the Two Strategies Applied Jointly (*Q7*)

In previous experiments, we validated the effectiveness of the token pruning strategy in reducing token usage and the query boosting strategy in improving task performance. Here, we explore the optimization effect of the two strategies applied jointly. For 1,000 node classification queries in each dataset, we first apply the token pruning strategy, sorting these queries in ascending order based on their text-inadequacy scores. The neighbor text of the top 20% of these sorted queries is omitted to reduce token usage. We then apply the query boosting strategy to execute these queries, which progressively enriches the neighbor text of unexecuted queries by incorporating pseudo-labels generated in earlier rounds.

Table VIII presents the results before and after optimization, where "w/ prune & boost" indicating the joint version. For token cost, we use "# Queries Equip $\mathcal{N}_i$" as an indicator: a higher number of queries equipping $\mathcal{N}_i$ results in higher token cost. Compared to the original counterparts, the joint version not only removes the token cost incurred by the neighbor text of 20% queries, but also achieves higher accuracy in most scenarios. This demonstrates that the joint application of our two optimization strategies can simultaneously minimize token costs and enhance task performance, offering a comprehensive improvement for benchmark methods.

TABLE IX
COMPARISON OF CLASSIFICATION ACCURACY (%) ON THE CORA DATASET AFTER APPLYING OUR OPTIMIZATION STRATEGIES TO INSTRUCTION TUNING-BASED METHODS.

Backbone	Base	w/ boost	w/ random	w/ prune	w/ both
1-hop, w/ raw, no path	84.2	85.8	78.6	83.1	84.2
2-hop, w/ raw, no path	84.4	86.2	78.4	83.9	85.2
2-hop, w/ raw, w/ path	84.6	85.8	78.9	83.9	84.3
1-hop, no raw, no path	73.3	74.7	71.1	74.9	75.5
2-hop, no raw, no path	83.3	84.2	78.2	82.4	83.5
2-hop, no raw, w/ path	82.8	85.5	77.4	81.9	84.5

TABLE X
COMPARISON OF LINK PREDICTION ACCURACY (%) BEFORE AND AFTER OPTIMIZATION. ↑ DENOTES IMPROVEMENT.

Dataset	Vanilla	Base	w/ boost	w/ prune	w/ both
Cora	73.0	76.1	80.2↑	75.8	79.0↑
Citeseer	86.8	88.4	89.6↑	88.5	89.8↑
Pubmed	87.5	86.9	88.3↑	87.3	88.1↑

*I. Effectiveness of the Two Strategies on Instruction Tuning-Based Methods (**Q8**)*

Beyond black-box LLM experiments, this section examines our strategies for instruction tuning-based "LLMs as predictors" methods. These methods typically align graph tokens with language tokens through training. Some variants substitute raw text of neighbors with their corresponding graph tokens. However, this does not affect our token pruning, as the type of token—whether graph or language—does not alter the pruning process. We experimented with six backbones from instructGLM [44], covering various configurations such as the inclusion of raw neighbor text (w/ raw vs. no raw), the number of hops considered (1-hop vs. 2-hop), and whether neighbor path descriptions are used (w/ path vs. no path). We evaluated five configurations for each backbone: 1) the unchanged model (Base), 2) with query boosting (w/ boost), 3) with random pruning (w/ random), 4) with our token pruning (w/ prune), and 5) with both token pruning and query boosting (w/ both). In the last three variations, 30% of the queries had their neighbors pruned to reduce token costs.

As shown in Table IX, our strategies consistently deliver expected results across different backbones. The higher accuracy of "w/ prune" compared to "w/ random" demonstrates the trade-off advantage of our token pruning, achieving token reduction with much less accuracy loss. Furthermore, the accuracy gains of "w/ boost" over "Base" and "w/ both" over "w/ prune" confirm the effectiveness of our query boosting. This experiment validates that our strategies are equally applicable and effective for instruction tuning-based methods.

*J. Effectiveness of Our Strategies in Link Prediction Task (**Q9**)*

This section investigates the effectiveness of our strategies for link prediction, which predicts the existence of an edge between a node pair. Since category information is not involved in link prediction, the token pruning strategy computes a node pair's text inadequacy directly from the surrogate binary classifier f_{θ^*}'s output confidence (i.e., the maximum probability value), such that $\mathrm{D}(t_i, t_j) = 1 - \max(f_{\theta^*}(x_i||x_j))$. In query boosting, selecting the candidate set $\mathcal{C}$ no longer needs to consider conflicting labels, so $\mathcal{C} = \{v_i \mid |\mathcal{N}_i| \geq \gamma_1\}$. The operational flow is largely consistent with node classification. For each dataset, we evaluated: 1) Vanilla: Node pair text alone; 2) Base: Node pair text with neighbor links; 3) w/ boost: Enriches prompts with new neighbors via query boosting; 4) w/ prune: Prunes 20% of test edges' neighbor links; 5) w/ both: Prunes 20% of edges and boosts with new neighbors.

In Table X, we show that our optimization strategies remain effective in the link prediction task. The "w/ prune" version maintains accuracy similar to the "Base" version, indicating that its approach for measuring text adequacy is also suitable for node pairs. Meanwhile, the accuracy improvement in the "w/ boost" version indicates that the inclusion of pseudo-labels from neighboring queries is also effective for link prediction, further confirming the generality of our query boosting strategy. This experiment highlights the versatility of our strategies across node-level tasks.

VII. CONCLUSION

Building on an information-theoretic analysis at the single query level, this paper proposes two strategies to optimize the execution of multiple node queries with LLMs: token pruning and query boosting. The token pruning strategy reduces token usage of existing methods without compromising task performance, while the query boosting strategy enhances task performance with negligible additional token cost. Comprehensive experiments validate the effectiveness, universality, and application potential of both strategies. Their advantages are particularly pronounced in large-scale query scenarios, excelling in token savings and effectively utilizing neighbor pseudo-labels. These strategies significantly improve the cost-efficiency and practicality of deploying LLMs on graph-related tasks, especially in resource-intensive settings.

These optimization strategies are applicable across various "LLMs as predictors" frameworks, whether black-box LLMs or instruction tuning-based approaches. They are also scalable to different node-level tasks; however, they are not yet directly applicable to graph-level tasks, such as graph classification. Future work could explore extending these strategies to graph-level tasks, such as refining token pruning to exclude irrelevant subgraph tokens in these contexts.

ACKNOWLEDGMENT

This work was partially supported by the National Natural Science Foundation of China (NSFC) under Grants 92270125 and 62276024, the National Research Foundation, Singapore, under its National Large Language Models Funding Initiative (AISG Award No. AISG-NMLP-2024-004), and the China Scholarship Council No. 202306030110. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

REFERENCES

[1] A. Li, B. Yang, H. Huo, F. K. Hussain, and G. Xu, "Structure- and logic-aware heterogeneous graph learning for recommendation," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024, pp. 544–556.

[2] C. Wu, C. Wang, J. Xu, Z. Fang, T. Gu, C. Wang, Y. Song, K. Zheng, X. Wang, and G. Zhou, "Instant representation learning for recommendation over large dynamic graphs," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. Los Alamitos, CA, USA: IEEE Computer Society, April 2023, pp. 82–95.

[3] S. Gurukar, N. Pancha, A. Zhai, E. Kim, S. Hu, S. Parthasarathy, C. Rosenberg, and J. Leskovec, "Multibisage: A web-scale recommendation system using multiple bipartite graphs at pinterest," *Proc. VLDB Endow.*, vol. 16, no. 4, p. 781–789, Dec. 2022.

[4] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ser. CIKM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 315–324.

[5] D. Cheng, X. Wang, Y. Zhang, and L. Zhang, "Graph neural network for fraud detection via spatial-temporal attention," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 08, pp. 3800–3813, August 2022.

[6] J. Jiang, Y. Li, B. He, B. Hooi, J. Chen, and J. K. Z. Kang, "Spade: A real-time fraud detection framework on evolving graphs," *Proc. VLDB Endow.*, vol. 16, no. 3, p. 461–469, Nov. 2022.

[7] K. M. Saifuddin, B. Bumgardnerr, F. Tanvir, and E. Akbas, "Hygnn: Drug-drug interaction prediction via hypergraph neural network," *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pp. 1503–1516, 2022.

[8] J. M. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, pp. 583 – 589, 2021.

[9] J. Liu, G. Tan, W. Lan, and J. Wang, "Identification of early mild cognitive impairment using multi-modal data and graph convolutional networks," *BMC Bioinformatics*, vol. 21, 11 2020.

[10] J. Peng, Z. Chen, Y. Shao, Y. Shen, L. Chen, and J. Cao, "Sancus: staleness-aware communication-avoiding full-graph decentralized training in large-scale graph neural networks," *Proc. VLDB Endow.*, vol. 15, no. 9, p. 1937–1950, May 2022.

[11] X. Gao, T. Chen, Y. Zang, W. Zhang, Q. V. Hung Nguyen, K. Zheng, and H. Yin, " Graph Condensation for Inductive Node Representation Learning ," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2024, pp. 3056–3069.

[12] Y. Xu, B. Shi, T. Ma, B. Dong, H. Zhou, and Q. Zheng, "Cldg: Contrastive learning on dynamic graphs," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 2023, pp. 696–707.

[13] J. Zhao, H. Mostafa, M. Galkin, M. Bronstein, Z. Zhu, and J. Tang, "Graphany: A foundation model for node classification on any graph," 2024. [Online]. Available: https://arxiv.org/abs/2405.20445

[14] G. V. Demirci, A. Haldar, and H. Ferhatosmanoglu, "Scalable graph convolutional network training on distributed-memory systems," *Proc. VLDB Endow.*, vol. 16, no. 4, p. 711–724, Dec. 2022.

[15] X. Gao, W. Zhang, J. Yu, Y. Shao, Q. V. H. Nguyen, B. Cui, and H. Yin, "Accelerating scalable graph neural network inference with node-adaptive propagation," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 3042–3055.

[16] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.

[17] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graph-SAINT: Graph sampling based inductive learning method," in *International Conference on Learning Representations*, 2020.

[18] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep Graph Infomax," in *International Conference on Learning Representations*, 2019.

[19] R. Li, S. Di, L. Chen, and X. Zhou, "Gradgcl: Gradient graph contrastive learning," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024, pp. 1171–1184.

[20] G. Kusano, "Ga-tag: Data enrichment with an automatic tagging system utilizing large language models," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024, pp. 5397–5400.

[21] Y. Lou, C. Lei, X. Qin, Z. Wang, C. Faloutsos, R. Anubhai, and H. Rangwala, "Datalore: Can a large language model find all lost scrolls in a data repository?" in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024, pp. 5170–5176.

[22] M. Pourreza and D. Rafiei, "Din-sql: decomposed in-context learning of text-to-sql with self-correction," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23. Red Hook, NY, USA: Curran Associates Inc., 2024.

[23] T. Ren, Y. Fan, Z. He, R. Huang, J. Dai, C. Huang, Y. Jing, K. Zhang, Y. Yang, and X. S. Wang, " PURPLE: Making a Large Language Model a Better SQL Writer ," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2024, pp. 15–28.

[24] J.-P. Zhu, P. Cai, B. Niu, Z. Ni, K. Xu, J. Huang, J. Wan, S. Ma, B. Wang, D. Zhang, L. Tang, and Q. Liu, "Chat2query: A zero-shot automatic exploratory data analysis system with large language models," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, 2024, pp. 5429–5432.

[25] Z. Chen, H. Mao, H. Li, W. Jin, H. Wen, X. Wei, S. Wang, D. Yin, W. Fan, H. Liu, and J. Tang, "Exploring the potential of large language models (llms)in learning on graphs," *SIGKDD Explor.*, vol. 25, no. 2, pp. 42–61, 2023.

[26] J. Huang, X. Zhang, Q. Mei, and J. Ma, "Can LLMs effectively leverage graph structural information through prompts, and why?" *Transactions on Machine Learning Research*, 2024.

[27] R. Li, J. Li, J. Han, and G. Wang, "Similarity-based neighbor selection for graph llms," *arXiv preprint arXiv:2402.03720*, 2024.

[28] N. N. Dalvi, S. K. Sanghai, P. Roy, and S. Sudarshan, "Pipelining in multi-query optimization," in *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 59–70.

[29] S. Finkelstein, "Common expression analysis in database applications," in *Proceedings of the 1982 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '82. New York, NY, USA: Association for Computing Machinery, 1982, p. 235–245.

[30] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe, "Efficient and extensible algorithms for multi query optimization," *SIGMOD Rec.*, vol. 29, no. 2, p. 249–260, May 2000.

[31] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, ser. SOSP '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 611–626.

[32] J. Juravsky, B. Brown, R. Ehrlich, D. Y. Fu, C. Ré, and A. Mirhoseini, "Hydragen: High-throughput llm inference with shared prefixes," 2024.

[33] Z. Ye, R. Lai, B.-R. Lu, C.-Y. Lin, S. Zheng, L. Chen, T. Chen, and L. Ceze, "Cascade inference: Memory bandwidth efficient shared prefix batch decoding," February 2024.

[34] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual Review of Sociology*, vol. 27, no. Volume 27, 2001, pp. 415–444, 2001.

[35] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[36] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 1025–1035.

[37] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[38] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[39] H. Yan, C. Li, R. Long, C. Yan, J. Zhao, W. Zhuang, J. Yin, P. Zhang, W. Han, H. Sun, W. Deng, Q. Zhang, L. Sun, X. Xie, and S. Wang,

"A comprehensive study on text-attributed graphs: Benchmarking and rethinking," in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[40] Z. Chen, H. Mao, H. Wen, H. Han, W. Jin, H. Zhang, H. Liu, and J. Tang, "Label-free node classification on graphs with large language models (LLMs)," in *The Twelfth International Conference on Learning Representations*, 2024.

[41] X. He, X. Bresson, T. Laurent, A. Perold, Y. LeCun, and B. Hooi, "Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning," in *The Twelfth International Conference on Learning Representations*, 2024.

[42] S. Sun, Y. Ren, C. Ma, and X. Zhang, "Large language models as topological structure enhancers for text-attributed graphs," *arXiv preprint arXiv:2311.14324*, 2023.

[43] Z. Guo, L. Xia, Y. Yu, Y. Wang, Z. Yang, W. Wei, L. Pang, T. Chua, and C. Huang, "Graphedit: Large language models for graph structure learning," *CoRR*, vol. abs/2402.15183, 2024.

[44] R. Ye, C. Zhang, R. Wang, S. Xu, and Y. Zhang, "Language is all a graph needs," in *Findings of the Association for Computational Linguistics: EACL 2024*. St. Julian's, Malta: Association for Computational Linguistics, Mar. 2024, pp. 1955–1973.

[45] J. Tang, Y. Yang, W. Wei, L. Shi, L. Su, S. Cheng, D. Yin, and C. 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*, ser. SIGIR '24, 2024, p. 491–500.

[46] Z. Liu, X. He, Y. Tian, and N. V. Chawla, "Can we soft prompt llms for graph learning tasks?" in *Companion Proceedings of the ACM Web Conference 2024*, ser. WWW '24. Association for Computing Machinery, 2024, p. 481–484.

[47] T. K. Sellis, "Multiple-query optimization," *ACM Trans. Database Syst.*, vol. 13, no. 1, p. 23–52, Mar. 1988.

[48] M. Zhang, Z. Ji, Z. Luo, Y. Wu, and C. Chai, " Applications and Challenges for Large Language Models: From Data Management Perspective ," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2024, pp. 5530–5541.

[49] S. Liu, A. Biswal, A. Cheng, X. Mo, S. Cao, J. E. Gonzalez, I. Stoica, and M. Zaharia, "Optimizing llm queries in relational workloads," 2024.

[50] N. Bertschinger, J. Rauh, E. Olbrich, J. Jost, and N. Ay, "Quantifying unique information," *Entropy*, vol. 16, no. 4, pp. 2161–2183, 2014.

[51] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, pp. 127–163, 2000.

[52] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the third ACM conference on Digital libraries*, 1998, pp. 89–98.

[53] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[54] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020.

[55] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 6894–6910.