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

Yujie Fang[1], Xin Li[1*], Yuangang Pan[2], Xin Huang[3], Ivor W. Tsang[2]

[1] Beijing Institute of Technology

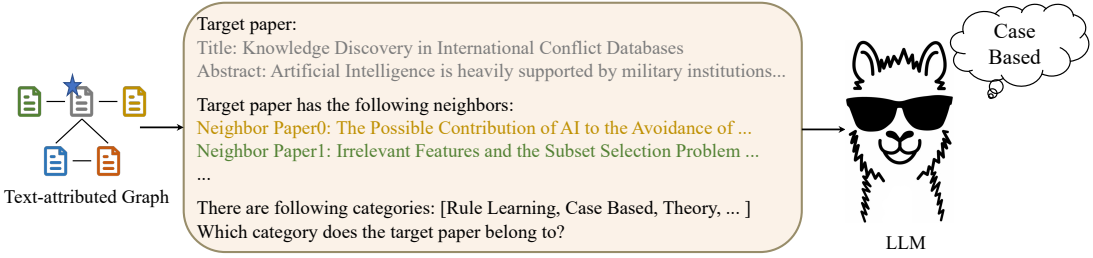
[2] Agency for Science, Technology and Research

[3] Hong Kong Baptist University

ICDE 2025, May 21, 2025

The Rise of LLMs in Graph AI

LLMs are becoming very popular. And they are being used for graph tasks, too.

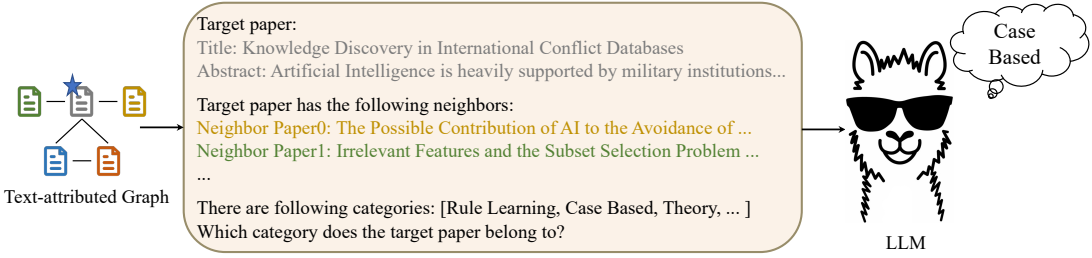


Example: An LLM predicts a node's category using its own text and the text of its neighbors.

This is a powerful new way to solve graph tasks !

The Rise of LLMs in Graph AI

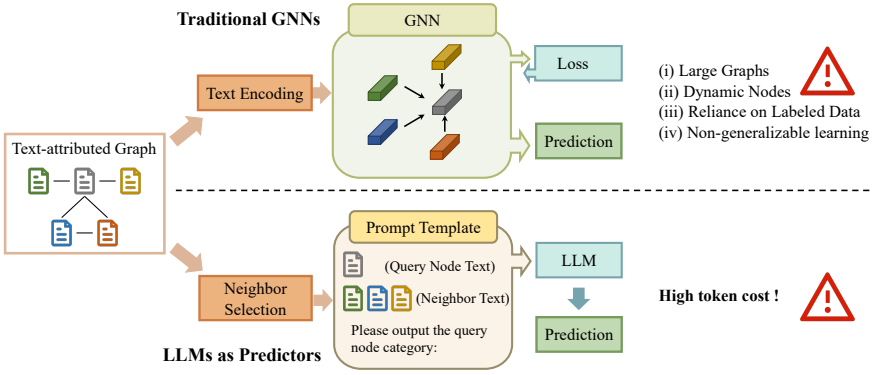
LLMs are becoming very popular. And they are being used for graph tasks, too.



Example: An LLM predicts a node's category using its own text and the text of its neighbors.

This is a powerful new way to solve graph tasks !

LLMs vs. GNNs



While LLMs offer some clear advantages over traditional GNNs，they have a major drawback: high token cost.

The Token Dilemma: Soaring Costs

Target paper:
Title: Knowledge Discovery in International Conflict Databases
Abstract: Artificial Intelligence is heavily supported by military institutions...

Target paper has the following neighbors:
Neighbor Paper0: The Possible Contribution of AI to the Avoidance of ...
Neighbor Paper1: Irrelevant Features and the Subset Selection Problem ...
Neighbor Paper2: Using Decision Trees to Improve Case-Based Learning ...
Neighbor Paper3: Feature Selection via Mathematical Programming ...

There are following categories: [Rule Learning, Case Based, Theory, ...]
Which category does the target paper belong to?

- **The Neighbor Problem:**

  Each query includes the text from the target node and its neighbors.

  And neighbor text can be very long, especially if the node has many neighbors.

Key Question: With a limited token budget, how do we handle large-scale graph queries efficiently?

The Token Dilemma: Soaring Costs

- **Pay-Per-Token:**

  Companies like OpenAI charge you for each token(e.g., $30 per million).

  Every single word and symbol costs money.

Key Question: With a limited token budget, how do we handle large-scale graph queries efficiently?

The Token Dilemma: Soaring Costs

- **Prohibitive Costs at Scale:**

  When processing millions of queries, costs can exceed $300K.

  This makes large-scale deployment of LLMs impractical without optimization.

Key Question: With a limited token budget, how do we handle large-scale graph queries efficiently?

Our Contributions

We introduce two Multi-Query Optimization (MQO) strategies for LLMs on graphs:

Token Pruning

- Reduces token usage of existing methods without degrading prediction accuracy.
- Provides a cost-efficient way to allocate tokens when the token budget is tight.

Query Boosting

- Improves accuracy by adding pseudo-labels from already processed, connected queries.
- Highly cost-effective, as the added pseudo-labels are very short.

Making LLMs more cost-efficient for graph tasks!

Our Goal: Multi-Query Optimization (MQO)

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Multi-Query Optimization (MQO):

- Objective: Construct an execution plan π for a query set $\mathcal{V}_Q$ that maximizes task performance under a limited token budget B.
- Mathematically:

$$\pi^* = \arg\max_{\pi} \sum_{v_i \in \mathcal{V}_Q} \mathbf{1}(y_i = \hat{y}_i)$$

$$\text{s.t.} \sum_{v_i \in \mathcal{V}_Q} \text{Tokens}(\pi \circ v_i) \leq B$$

- Where $\mathbf{1}(\cdot)$ is the indicator function, and $\text{Tokens}(\pi \circ v_i)$ is the token cost of query v_i under plan π.

Single Query Formulation:

- An LLM predicts label $\hat{y}_i$ for node v_i:

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

  - t_i: Text of the target node v_i.
  - $\mathcal{N}_i$: Selected neighbor text.
  - prompt: Task instruction.

Neighbor Text Examples $\mathcal{N}_i$:

Method	Strategy for $\mathcal{N}_i$
1-hop random	Text from random 1-hop neighbors
2-hop random	Text from random 2-hop neighbors
SNS	Explore hops for labeled neighbors, then rank by similarity.

Our Goal: Multi-Query Optimization (MQO)

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Multi-Query Optimization (MQO):

- Objective: Construct an execution plan π for a query set $\mathcal{V}_Q$ that maximizes task performance under a limited token budget B.
- Mathematically:

$$\pi^* = \arg\max_{\pi} \sum_{v_i \in \mathcal{V}_Q} \mathbf{1}(y_i = \hat{y}_i)$$

$$\text{s.t.} \sum_{v_i \in \mathcal{V}_Q} \text{Tokens}(\pi \circ v_i) \leq B$$

- Where $\mathbf{1}(\cdot)$ is the indicator function, and $\text{Tokens}(\pi \circ v_i)$ is the token cost of query v_i under plan π.

Single Query Formulation:

- An LLM predicts label $\hat{y}_i$ for node v_i:

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

  - t_i: Text of the target node v_i.
  - $\mathcal{N}_i$: Selected neighbor text.
  - prompt: Task instruction.

Neighbor Text Examples $\mathcal{N}_i$:

Method	Strategy for $\mathcal{N}_i$
1-hop random	Text from random 1-hop neighbors
2-hop random	Text from random 2-hop neighbors
SNS	Explore hops for labeled neighbors, then rank by similarity.

Our Goal: Multi-Query Optimization (MQO)

Multi-Query Optimization (MQO):

- Objective: Construct an execution plan π for a query set $\mathcal{V}_Q$ that maximizes task performance under a limited token budget B.
- Mathematically:

$$\pi^* = \arg\max_{\pi} \sum_{v_i \in \mathcal{V}_Q} \mathbf{1}(y_i = \hat{y}_i)$$

$$\text{s.t.} \sum_{v_i \in \mathcal{V}_Q} \text{Tokens}(\pi \circ v_i) \leq B$$

- Where $\mathbf{1}(\cdot)$ is the indicator function, and $\text{Tokens}(\pi \circ v_i)$ is the token cost of query v_i under plan π.

Single Query Formulation:

- An LLM predicts label $\hat{y}_i$ for node v_i:

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

  - t_i: Text of the target node v_i.
  - $\mathcal{N}_i$: Selected neighbor text.
  - prompt: Task instruction.

Neighbor Text Examples $\mathcal{N}_i$:

Method	Strategy for $\mathcal{N}_i$
1-hop random	Text from random 1-hop neighbors
2-hop random	Text from random 2-hop neighbors
SNS	Explore hops for labeled neighbors, then rank by similarity.

Costly Neighbor Text Analysis

Does sending neighbor text always help?

Costly Neighbor Text Analysis

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Does sending neighbor text always help? We use mutual information to figure this out.

Target paper:
Title: Knowledge Discovery in International Conflict Databases
Abstract: Artificial Intelligence is heavily supported by military institutions...

There are following categories: [Rule Learning, Case Based, Theory, ...]
Which category does the target paper belong to?

Query with target node text t_i only: $\mathrm{I}(t_i; y_i)$

Query with t_i and neighbor text $\mathcal{N}_i$: $\mathrm{I}(t_i, \mathcal{N}_i; y_i) \Longrightarrow$

Target paper:
Title: Knowledge Discovery in International Conflict Databases
Abstract: Artificial Intelligence is heavily supported by military institutions...

Target paper has the following neighbors:
Neighbor Paper0: The Possible Contribution of AI to the Avoidance of ...
Neighbor Paper1: Irrelevant Features and the Subset Selection Problem ...
Neighbor Paper2: Using Decision Trees to Improve Case-Based Learning ...
Neighbor Paper3: Feature Selection via Mathematical Programming ...

There are following categories: [Rule Learning, Case Based, Theory, ...]
Which category does the target paper belong to?

Costly Neighbor Text Analysis

Does sending neighbor text always help? We use mutual information to figure this out.

Target paper:
Title: Knowledge Discovery in International Conflict Databases
Abstract: Artificial Intelligence is heavily supported by military institutions...

There are following categories: [Rule Learning, Case Based, Theory, ...]
Which category does the target paper belong to?

Query with target node text t_i only: $\mathrm{I}(t_i; y_i)$

Query with t_i and neighbor text $\mathcal{N}_i$: $\mathrm{I}(t_i, \mathcal{N}_i; y_i) \Longrightarrow$

Target paper:
Title: Knowledge Discovery in International Conflict Databases
Abstract: Artificial Intelligence is heavily supported by military institutions...

Target paper has the following neighbors:
Neighbor Paper0: The Possible Contribution of AI to the Avoidance of ...
Neighbor Paper1: Irrelevant Features and the Subset Selection Problem ...
Neighbor Paper2: Using Decision Trees to Improve Case-Based Learning ...
Neighbor Paper3: Feature Selection via Mathematical Programming ...

There are following categories: [Rule Learning, Case Based, Theory, ...]
Which category does the target paper belong to?

Neighbor Information Gain ($\mathrm{IG}^{\mathcal{N}_i}$)

The extra useful information added by the neighbor text $\mathcal{N}_i$ is:

$$\mathrm{IG}^{\mathcal{N}_i} = \mathrm{I}(t_i, \mathcal{N}_i; y_i) - \mathrm{I}(t_i; y_i)$$

If $\mathrm{I}(t_i; y_i) = \mathrm{H}(y_i)$, then $\mathrm{IG}^{\mathcal{N}_i} = 0$ regardless of $\mathcal{N}_i$. And a larger $\mathrm{IG}^{\mathcal{N}_i}$ indicates $\mathcal{N}_i$ is more valuable.

Insight: Saturated vs. Non-Saturated Nodes

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Definition (Node Saturation)

- **Saturated Node:** The node's own text t_i is enough to figure out its category. Sending neighbor text for these nodes is a waste of money.
- **Non-Saturated Node:** The node's text t_i isn't quite enough. Neighbor text $\mathcal{N}_i$ *might* be helpful.

Information Gain Principle for MQO

To optimize predictions and minimize token cost for each query v_i:

$$\mathcal{N}_i^* = \begin{cases} \varnothing & \text{if } v_i \text{ is saturated (Save tokens!)} \\ \arg\max_{\mathcal{N}_i} \mathrm{IG}^{\mathcal{N}_i} & \text{if } v_i \text{ is non-saturated (Maximize information!)} \end{cases}$$

Insight: Saturated vs. Non-Saturated Nodes

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Definition (Node Saturation)

- **Saturated Node:** The node's own text t_i is enough to figure out its category. Sending neighbor text for these nodes is a waste of money.
- **Non-Saturated Node:** The node's text t_i isn't quite enough. Neighbor text $\mathcal{N}_i$ *might* be helpful.

Information Gain Principle for MQO

To optimize predictions and minimize token cost for each query v_i:

$$\mathcal{N}_i^* = \begin{cases} \varnothing & \text{if } v_i \text{ is saturated (Save tokens!)} \\ \arg\max_{\mathcal{N}_i} \mathrm{IG}^{\mathcal{N}_i} & \text{if } v_i \text{ is non-saturated (Maximize information!)} \end{cases}$$

Strategy 1: Token Pruning

Objective: Stop sending neighbor text for "saturated" nodes to save tokens.

Challenge: How do we find these "saturated" nodes from the query set $\mathcal{V}_Q$?

Solution: Text Inadequacy Score $\mathrm{D}(t_i)$

We use $\mathrm{D}(t_i)$ as a proxy to assess if t_i alone is sufficient for prediction. It considers:

- **Text Ambiguity:** How uncertain a simple model is about the node's category based only on its text t_i: $\mathrm{H}(f^*(t_i))$.
- **LLM Category Bias:** Does the LLM tends to get this type of node wrong: $b_i = f^*(t_i) \times w^{\mathrm{T}}$.
- **Integration:** These are combined: $\mathrm{D}(t_i) = g_{\theta_2^*}(\mathrm{H}(f^*(t_i)) \parallel b_i)$.

Key Idea: Lower $\mathrm{D}(t_i) \implies t_i$ is likely saturated $\implies$ Prune its neighbor text $\mathcal{N}_i$!

Strategy 1: Token Pruning

Objective: Stop sending neighbor text for "saturated" nodes to save tokens.

Challenge: How do we find these "saturated" nodes from the query set $\mathcal{V}_Q$?

Solution: Text Inadequacy Score $\mathrm{D}(t_i)$

We use $\mathrm{D}(t_i)$ as a proxy to assess if t_i alone is sufficient for prediction. It considers:

- **Text Ambiguity:** How uncertain a simple model is about the node's category based only on its text t_i: $\mathrm{H}(f^*(t_i))$.
- **LLM Category Bias:** Does the LLM tends to get this type of node wrong: $b_i = f^*(t_i) \times \boldsymbol{w}^{\mathrm{T}}$.
- **Integration:** These are combined: $\mathrm{D}(t_i) = g_{\theta_2^*}(\mathrm{H}(f^*(t_i)) \parallel b_i)$.

Key Idea: Lower $\mathrm{D}(t_i) \implies t_i$ is likely saturated $\implies$ Prune its neighbor text $\mathcal{N}_i$!

Strategy 1: Token Pruning

Objective: Stop sending neighbor text for "saturated" nodes to save tokens.

Challenge: How do we find these "saturated" nodes from the query set $\mathcal{V}_Q$?

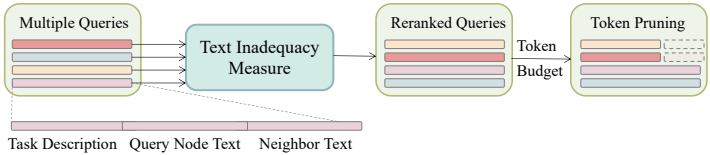
Solution: Text Inadequacy Score $\mathrm{D}(t_i)$

We use $\mathrm{D}(t_i)$ as a proxy to assess if t_i alone is sufficient for prediction. It considers:

- **Text Ambiguity:** How uncertain a simple model is about the node's category based only on its text t_i: $\mathrm{H}(f^*(t_i))$.
- **LLM Category Bias:** Does the LLM tends to get this type of node wrong: $b_i = f^*(t_i) \times \boldsymbol{w}^{\mathrm{T}}$.
- **Integration:** These are combined: $\mathrm{D}(t_i) = g_{\theta_2^*}(\mathrm{H}(f^*(t_i)) \parallel b_i)$.

Key Idea: Lower $\mathrm{D}(t_i)$ $\implies$ t_i is likely saturated $\implies$ Prune its neighbor text $\mathcal{N}_i$!

Customizable Token Allocation via Pruning



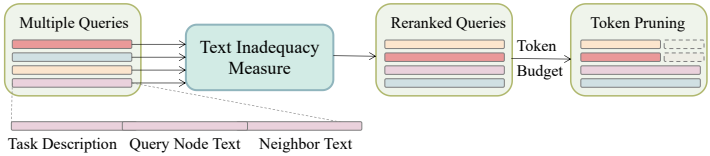
Procedure:

1. Calculate Text Inadequacy $\mathrm{D}(t_i)$ for each query in $\mathcal{V}_Q$.
2. Rank all the queries by this score, from lowest to highest; saturated nodes are expected at the top.
3. Based on the token budget, decide how many queries at the top of the list to prune.
4. For these top queries: use only t_i (i.e., $\mathcal{N}_i = \varnothing$).
5. For the rest: use t_i and selected $\mathcal{N}_i$.

Token Pruning helps us use our token budget smartly.
We spend tokens on neighbors only when they are most likely to help.

(Details in Algorithm 1 of the paper)

Customizable Token Allocation via Pruning



Procedure:

1. Calculate Text Inadequacy $\mathrm{D}(t_i)$ for each query in $\mathcal{V}_Q$.
2. Rank all the queries by this score, from lowest to highest; saturated nodes are expected at the top.
3. Based on the token budget, decide how many queries at the top of the list to prune.
4. For these top queries: use only t_i (i.e., $\mathcal{N}_i = \varnothing$).
5. For the rest: use t_i and selected $\mathcal{N}_i$.

Token Pruning helps us use our token budget smartly.
We spend tokens on neighbors only when they are most likely to help.

(Details in Algorithm 1 of the paper)

Strategy 2: Query Boosting

Objective: Improve prediction accuracy without adding many additional tokens.

Strategy 2: Query Boosting

Objective: Improve prediction accuracy without adding many additional tokens.

The pseudo-labels of neighbors are well-suited! These brief pseudo-labels consume minimal tokens and generally benefit predictions.

Strategy 2: Query Boosting

Objective: Improve prediction accuracy without adding many additional tokens.


The pseudo-labels of neighbors are well-suited! These brief pseudo-labels consume minimal tokens and generally benefit predictions.

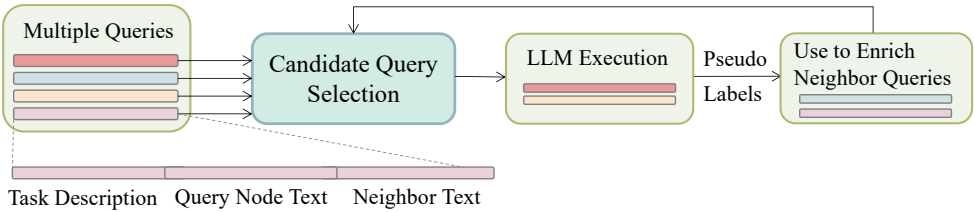
Our Idea: Enrich queries' neighbor text with pseudo-labels from already processed, connected queries.

Strategy 2: Query Boosting

Objective: Improve prediction accuracy without adding many additional tokens.

The pseudo-labels of neighbors are well-suited! These brief pseudo-labels consume minimal tokens and generally benefit predictions.

Our Idea: Enrich queries' neighbor text with pseudo-labels from already processed, connected queries.



Query Boosting: Round-Based Scheduling

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Challenge: What if those pseudo-labels are wrong? We don't want to spread errors!

Solution: Round-Based Query Scheduling

We process the easier, more certain ones first.

- **Earlier Rounds:** Queries with sufficient and consistent neighbor labels, as they are likely to receive correct pseudo-labels.
- **Later Rounds:** Queries with insufficient or conflicting neighbor labels. By waiting, they have a better chance to get enriched by pseudo-labels from the earlier rounds.
- The formal rule for selecting queries in each round:

$$\mathcal{C} = \left\{ v_i \,\middle|\, |\mathcal{N}_i^L| \geq \gamma_1 \ \wedge \ LC_i \leq \gamma_2 \right\}$$

- $|\mathcal{N}_i^L|$: Number of available (pseudo-)labeled neighbors.
- LC_i: Degree of label consistency/conflict among labeled neighbors.
- LLM predicts labels for $\mathcal{C}$; these become new pseudo-labels for remaining queries.

(Details in Algorithm 2 of the paper)

Query Boosting: Round-Based Scheduling

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Challenge: What if those pseudo-labels are wrong? We don't want to spread errors!

Solution: Round-Based Query Scheduling

We process the easier, more certain ones first.

- **Earlier Rounds:** Queries with sufficient and consistent neighbor labels, as they are likely to receive correct pseudo-labels.
- **Later Rounds:** Queries with insufficient or conflicting neighbor labels. By waiting, they have a better chance to get enriched by pseudo-labels from the earlier rounds.
- The formal rule for selecting queries in each round:

$$\mathcal{C} = \left\{ v_i \,\middle|\, |\mathcal{N}_i^L| \geq \gamma_1 \ \wedge \ LC_i \leq \gamma_2 \right\}$$

  - $|\mathcal{N}_i^L|$: Number of available (pseudo-)labeled neighbors.
  - LC_i: Degree of label consistency/conflict among labeled neighbors.
  - LLM predicts labels for $\mathcal{C}$; these become new pseudo-labels for remaining queries.

(Details in Algorithm 2 of the paper)

Query Boosting: Round-Based Scheduling

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Challenge: What if those pseudo-labels are wrong? We don't want to spread errors!

Solution: Round-Based Query Scheduling

We process the easier, more certain ones first.

- **Earlier Rounds:** Queries with sufficient and consistent neighbor labels, as they are likely to receive correct pseudo-labels.
- **Later Rounds:** Queries with insufficient or conflicting neighbor labels. By waiting, they have a better chance to get enriched by pseudo-labels from the earlier rounds.
- The formal rule for selecting queries in each round:

$$\mathcal{C} = \left\{ v_i \,\middle|\, |\mathcal{N}_i^L| \geq \gamma_1 \ \wedge \ LC_i \leq \gamma_2 \right\}$$

  - $|\mathcal{N}_i^L|$: Number of available (pseudo-)labeled neighbors.
  - LC_i: Degree of label consistency/conflict among labeled neighbors.
  - LLM predicts labels for $\mathcal{C}$; these become new pseudo-labels for remaining queries.

(Details in Algorithm 2 of the paper)

Query Boosting: Round-Based Scheduling

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Challenge: What if those pseudo-labels are wrong? We don't want to spread errors!

Solution: Round-Based Query Scheduling

We process the easier, more certain ones first.

- **Earlier Rounds:** Queries with sufficient and consistent neighbor labels, as they are likely to receive correct pseudo-labels.
- **Later Rounds:** Queries with insufficient or conflicting neighbor labels. By waiting, they have a better chance to get enriched by pseudo-labels from the earlier rounds.
- The formal rule for selecting queries in each round:

$$\mathcal{C} = \left\{ v_i \,\middle|\, |\mathcal{N}_i^L| \geq \gamma_1 \ \wedge \ LC_i \leq \gamma_2 \right\}$$

  - $|\mathcal{N}_i^L|$: Number of available (pseudo-)labeled neighbors.
  - LC_i: Degree of label consistency/conflict among labeled neighbors.
  - LLM predicts labels for $\mathcal{C}$; these become new pseudo-labels for remaining queries.

(Details in Algorithm 2 of the paper)

Query Boosting: Round-Based Scheduling

Challenge: What if those pseudo-labels are wrong? We don't want to spread errors!

Solution: Round-Based Query Scheduling

We process the easier, more certain ones first.

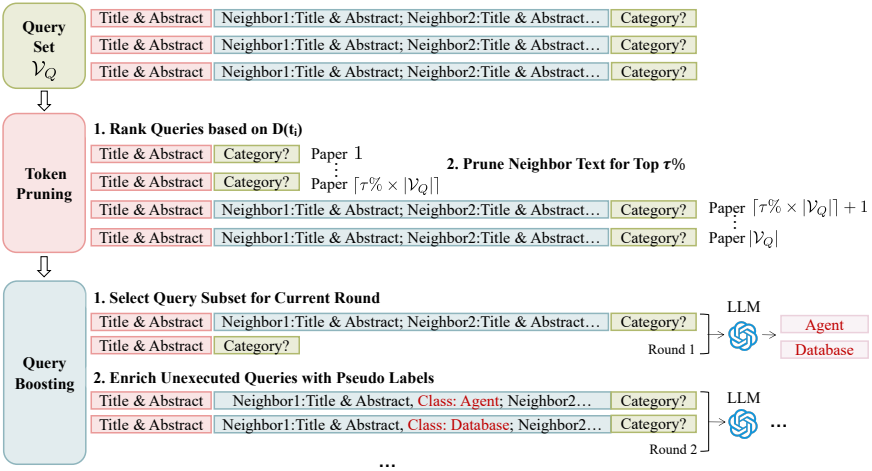
- **Earlier Rounds:** Queries with sufficient and consistent neighbor labels, as they are likely to receive correct pseudo-labels.
- **Later Rounds:** Queries with insufficient or conflicting neighbor labels. By waiting, they have a better chance to get enriched by pseudo-labels from the earlier rounds.
- The formal rule for selecting queries in each round:

$$\mathcal{C} = \left\{ v_i \,\middle|\, |\mathcal{N}_i^L| \geq \gamma_1 \ \wedge \ LC_i \leq \gamma_2 \right\}$$

- $|\mathcal{N}_i^L|$: Number of available (pseudo-)labeled neighbors.
- LC_i: Degree of label consistency/conflict among labeled neighbors.
- LLM predicts labels for $\mathcal{C}$; these become new pseudo-labels for remaining queries.

(Details in Algorithm 2 of the paper)

A Running Example: Pruning & Boosting



These strategies can be applied independently or combined as a two-step pipeline, reducing token usage while boosting accuracy.

Experiments: Effectiveness of Token Pruning

北京理工大学
BEIJING INSTITUTE OF TECHNOLOGY

Classification Accuracy Change (△%) after Token Pruning (Prune Top 20% Queries)

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

Finding 1: Applying our Token Pruning strategy by cutting out neighbor text for the top 20% of queries results in negligible accuracy degradation.

Experiments: Effectiveness of Token Pruning

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Classification Accuracy Change (△%) after Token Pruning (Prune Top 20% Queries)

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

Finding 1: Applying our Token Pruning strategy by cutting out neighbor text for the top 20% of queries results in negligible accuracy degradation.

Evaluate the performance of Token Pruning under a series of limited token budgets.

Finding 2: Token Pruning consistently results in higher accuracy compared to randomly selecting queries.

Experiments: Effectiveness of Token Pruning

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Classification Accuracy Change (△%) after Token Pruning (Prune Top 20% Queries)

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

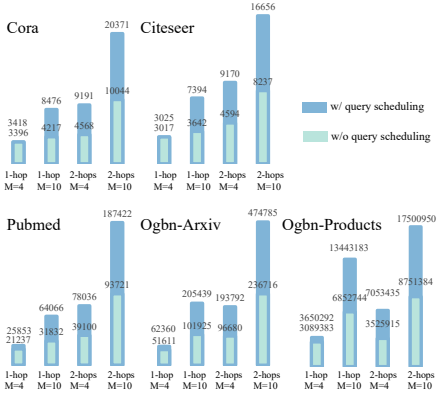
Finding 1: Applying our Token Pruning strategy by cutting out neighbor text for the top 20% of queries results in negligible accuracy degradation.

Evaluate the performance of Token Pruning under a series of limited token budgets.

Finding 2: Token Pruning consistently results in higher accuracy compared to randomly selecting queries.

Token Pruning works! It effectively optimizes token allocation when faced with limited budgets.

Experiments: Effectiveness of Query Boosting



Pseudo-label utilization frequencies of round-based query scheduling

Finding 3: Round-based query scheduling clearly increases pseudo-label utilization—nearly doubling it compared to a random order.

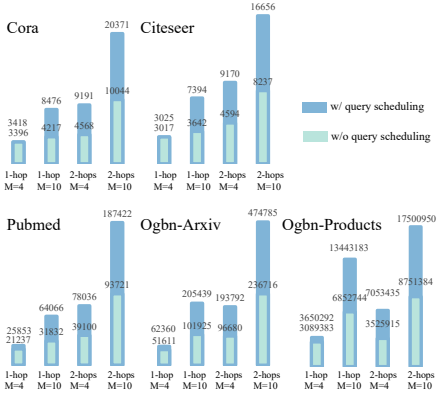
Classification Accuracy (%) Before and After Query Boosting

	GPT 4o-mini			GPT 3.5		
Method	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
1-hop random	67.8	61.0	79.4	72.3	64.1	87.4
+ 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
+ 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
+ Boost	72.6↑	70.6↑	81.6↑	76.3↑	70.6↑	90.3↑

Finding 4: Query Boosting consistently improves accuracy across different datasets and with different LLMs.

Query Boosting helps us improve prediction accuracy with very little token overhead (pseudo-labels are short).

Experiments: Effectiveness of Query Boosting



Pseudo-label utilization frequencies of round-based query scheduling

Finding 3: Round-based query scheduling clearly increases pseudo-label utilization—nearly doubling it compared to a random order.

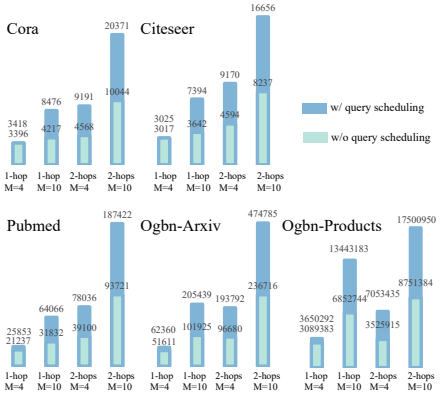
Classification Accuracy (%) Before and After Query Boosting

	GPT 4o-mini			*GPT 3.5*		
Method	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
1-hop random	67.8	61.0	79.4	72.3	64.1	87.4
+ 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
+ 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
+ Boost	72.6↑	70.6↑	81.6↑	76.3↑	70.6↑	90.3↑

Finding 4: Query Boosting consistently improves accuracy across different datasets and with different LLMs.

Query Boosting helps us improve prediction accuracy with very little token overhead (pseudo-labels are short).

Experiments: Effectiveness of Query Boosting



Pseudo-label utilization frequencies of round-based query scheduling

Finding 3: Round-based query scheduling clearly increases pseudo-label utilization—nearly doubling it compared to a random order.

Classification Accuracy (%) Before and After Query Boosting

	GPT 4o-mini			*GPT 3.5*		
Method	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
1-hop random	67.8	61.0	79.4	72.3	64.1	87.4
+ 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
+ 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
+ Boost	72.6↑	70.6↑	81.6↑	76.3↑	70.6↑	90.3↑

Finding 4: Query Boosting consistently improves accuracy across different datasets and with different LLMs.

Query Boosting helps us improve prediction accuracy with very little token overhead (pseudo-labels are short).

Experiments: Joint Application of Strategies

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

Classification Accuracy (%) and Token Cost (Queries with $\mathcal{N}_i$) for Joint Strategy.

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

Finding 5: The joint strategy (+ Prune & Boost) cuts down the number of queries with neighbor text, but still achieves higher accuracy in most cases.

Broad Applicability

- **Versatile across LLMs:** Works with both black-box and instruction-tuned LLMs.
- **Scalable to node-level tasks:** Also effective for link prediction: fewer tokens, better predictions.

Experiments: Joint Application of Strategies

Classification Accuracy (%) and Token Cost (Queries with $\mathcal{N}_i$) for Joint Strategy.

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

Finding 5: The joint strategy (+ Prune & Boost) cuts down the number of queries with neighbor text, but still achieves higher accuracy in most cases.

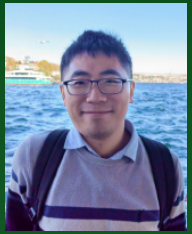
Broad Applicability

- **Versatile across LLMs:** Works with both black-box and instruction-tuned LLMs.
- **Scalable to node-level tasks:** Also effective for link prediction: fewer tokens, better predictions.

Thanks for your attention!

Special thanks to our collaborators!

Xin Li



Yuangang Pan

Xin Huang

Ivor W. Tsang

Let’s skip the on-stage Q&A.
I’m happy to discuss anything about our work or LLMs on graphs off-stage
—my nervous English might not do it justice here.
Thanks for understanding!