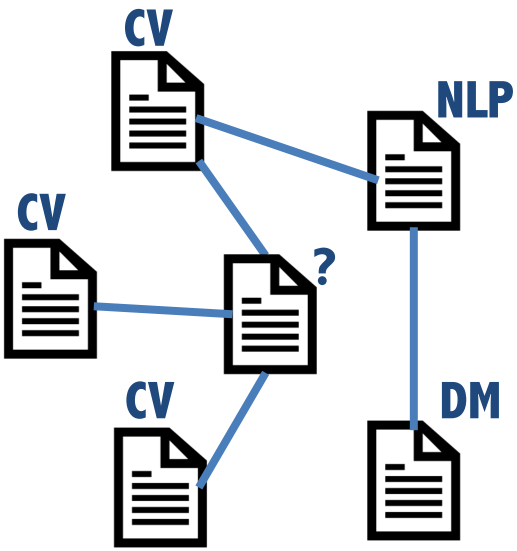
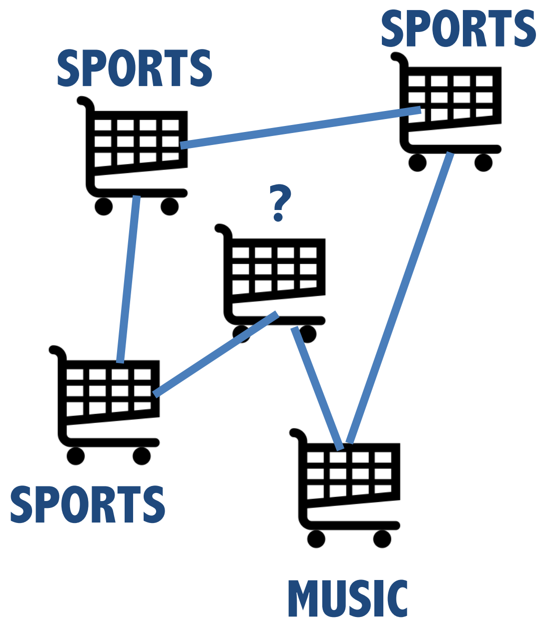
LABEL-FREE NODE CLASSIFICATION ON GRAPHS WITH LARGE LANGUAGE MODELS (LLMS)

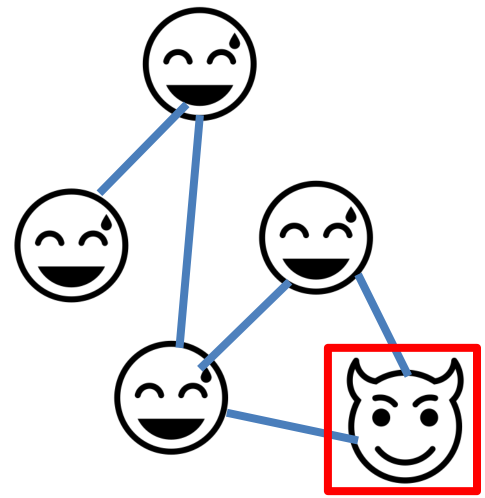
Node Classification is a crucial task for graph



Paper Categorization



Product Classification



Fraud Detection

Semi-supervised node classification on graphs

Semi-supervised node classification on graphs

Given a fixed training set

- Node features X
- Graph Structure A
- Ground truth labels y_L

Predict the labels of the rest nodes

Semi-supervised node classification on graphs

Given a fixed training set

- Node features X
- Graph Structure A
- Ground truth labels y_L

Predict the labels of the rest nodes

Graph neural networks work well for this task with **abundant ground truth labels**

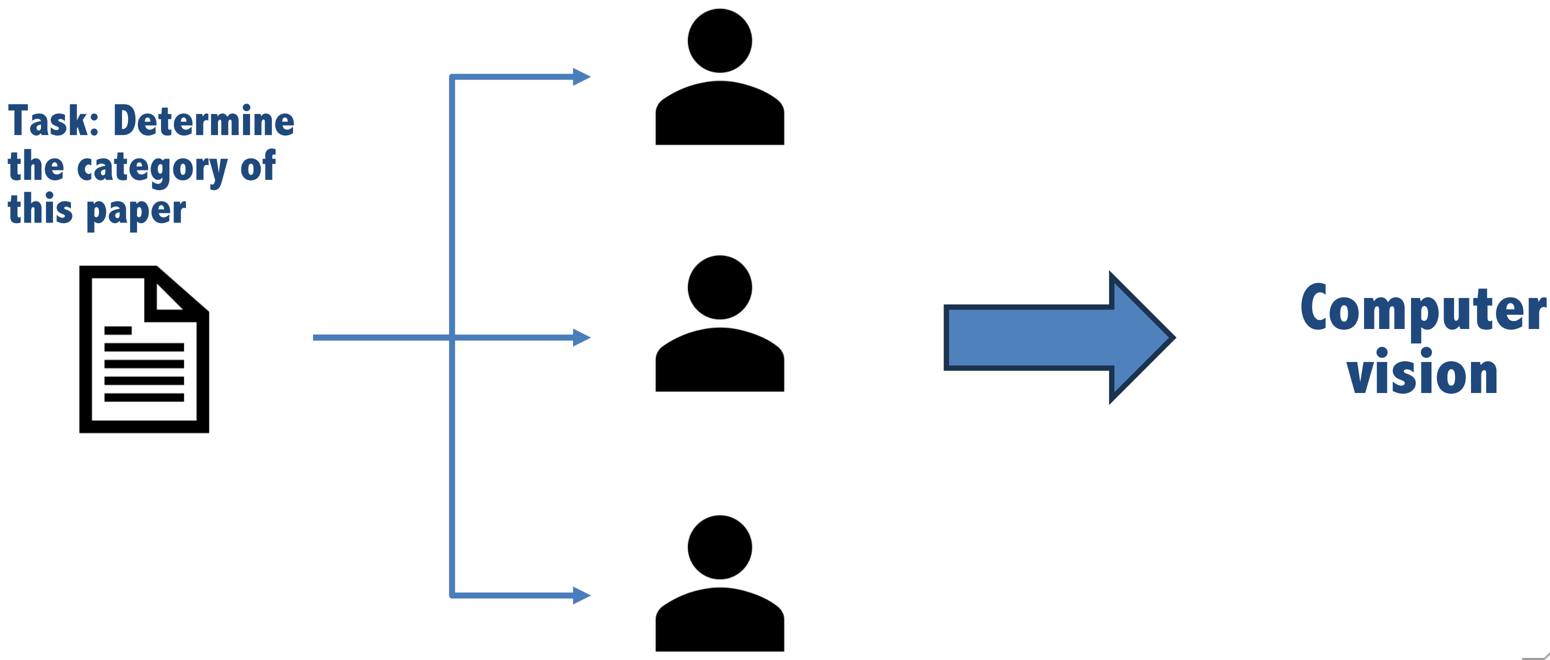
Two assumptions

A fixed training set → Overlook the data selection process

Ground truth labels y_L → Overlook the intricacy of (graph) data annotation

The old story: Human Annotation

Crowdsourcing platform (like Amazon MTurk) is one of the most popular ways to do annotations



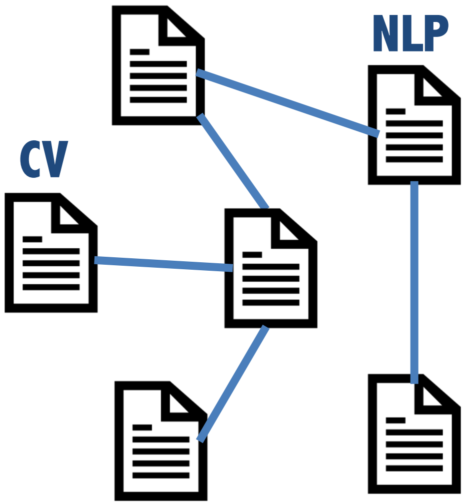
The old story: Human Annotation

Crowdsourcing platform (like Amazon MTurk) is one of the most popular ways to do annotations

How good is it?

Even for a simple task like annotating CIFAR-10 (image of daily objects), accuracy is only around 80%

Annotating graph data is challenging



- Due to the non-IID nature of the graph, human annotations tend to be **biased and focus on a small group of nodes***
- Annotating some kinds of graph, like OGB-Arxiv (paper), requires related knowledge
- Annotating a massive scale graph, like million-scale OGB-Products, needs lots of time and money

* Zhu, Qi, et al. "Shift-robust gnns: Overcoming the limitations of localized graph training data." *Advances in Neural Information Processing Systems* 34 (2021): 27965-27977.

LLMs as annotators for graphs?

In recent literature*, LLMs present promising zero-shot performance on node classification tasks

Limitations

- **Cannot utilize graph structure**
- **Performance gap to well-trained GNNs**
- **Expensive & slow for inference**

Using LLMs as annotators for GNNs seems a plausible way to harness the strength of both GNNs and LLMs!

Chen, Zhikai, et al. "Exploring the potential of large language models (llms) in learning on graphs." *arXiv preprint arXiv:2307.03393* (2023).

New challenges

Data Selection → LLMs → GNNs

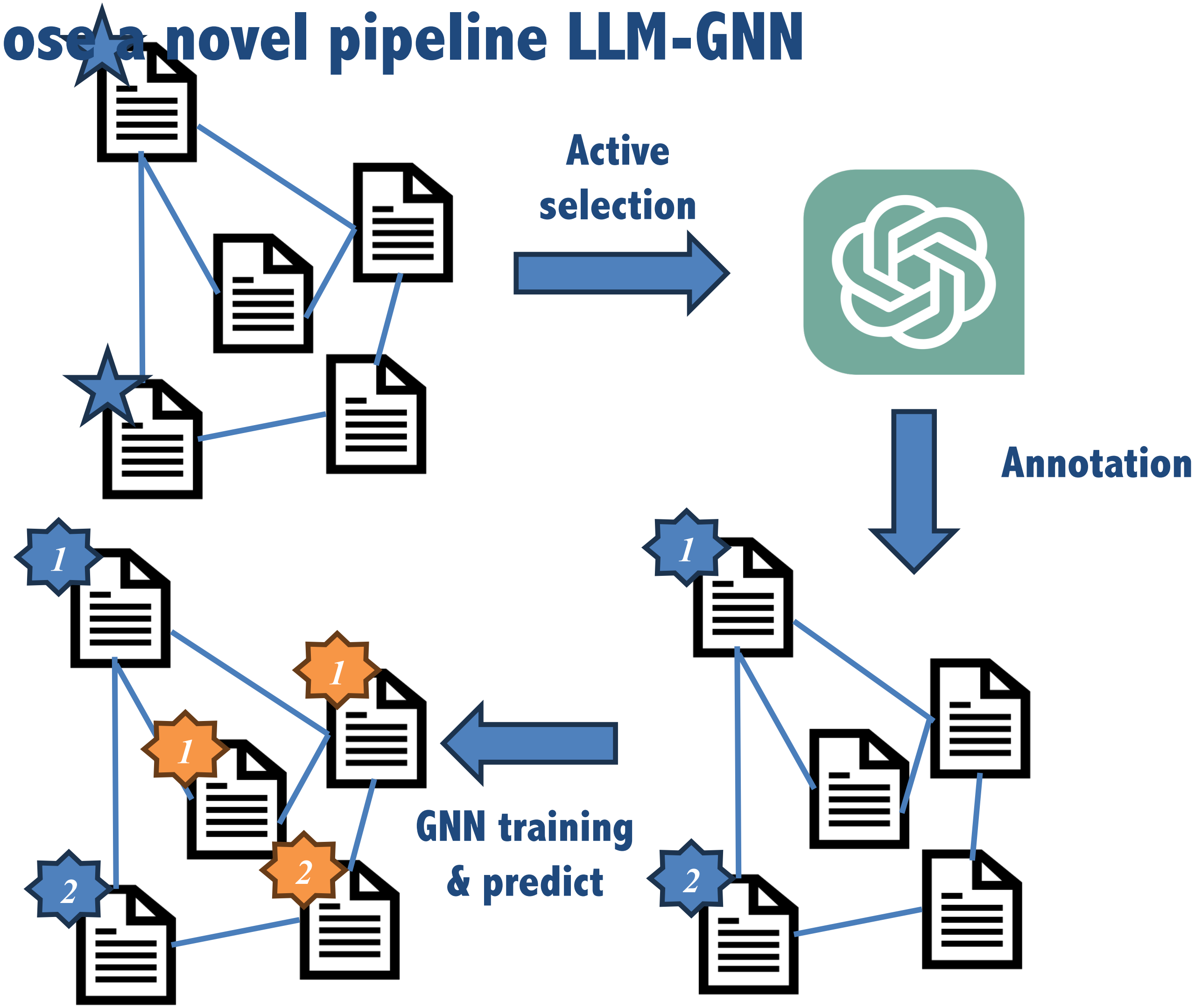
How can we ensure

- Diversity & Representativeness
  - **Critical for performance in graph active learning**
- High Annotation quality

→ Optimize the trained GNN performance

Label-free node classification on graphs with LLMs

We propose a novel pipeline LLM-GNN



Implementation

LLM-GNN supports flexible component design

The key part is how to consider the following two factors simultaneously (we show one possible implementation)

Diversity & Representativeness

➡ Can be addressed by graph active learning

Annotation quality

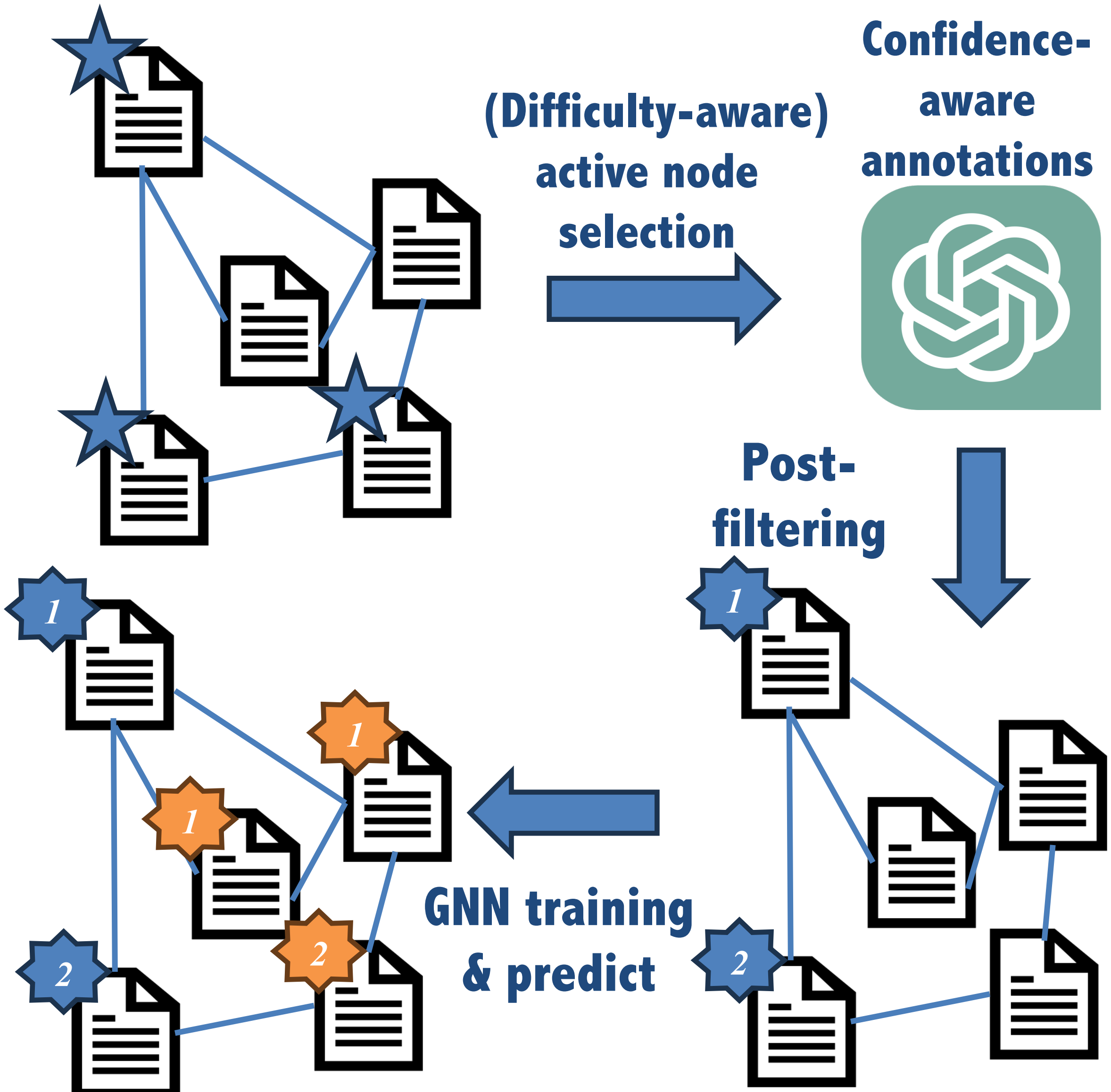
➡ We propose

1. Difficulty-aware active selection

2. Confidence-aware prompt + Post filtering

Implementation

LLM-related information is not available, heuristic-based methods



Use LLMs' outputs to further optimize the selection

Loss function for noisy label learning

Difficulty-aware active node selection

In the selection stage, only feature and structure is available

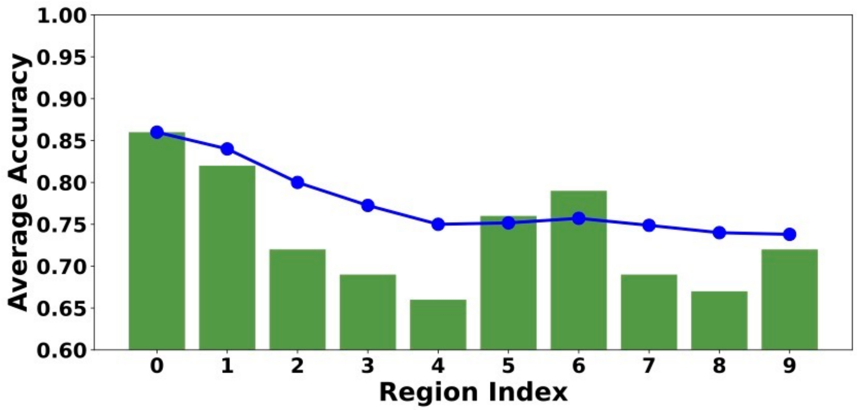
We induce the difficulty of annotation by the rule of thumb

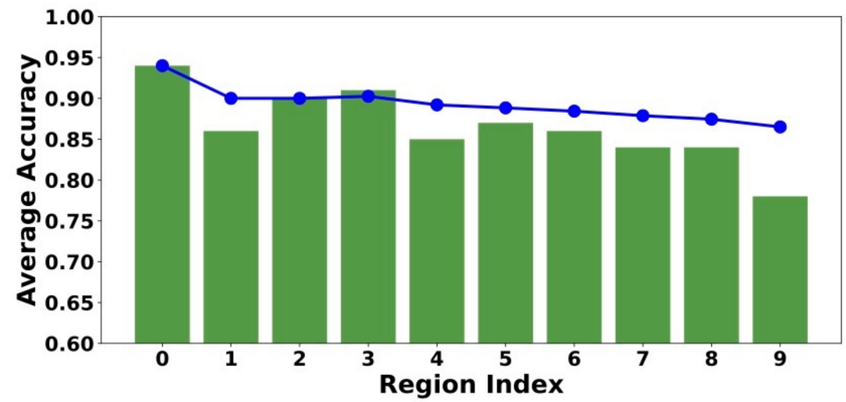
The difficulty of annotation can be induced from density of nodes in the feature space

Distance of nodes to their closest clustering centers (CC)

Difficulty-aware active node selection

If we group and sort nodes with their distances to each one's CC





LLMs present better annotation quality (lower difficulty) to those nodes closer to their CC

Intuition: Closer to CC indicating nodes with more "common" features, it may be easier for LLMs to annotate "common" nodes

Difficulty-aware active node selection

Our methods: Combining difficulty-aware metrics with traditional graph active learning metrics

$f_{act}(v_i)$

$$CDensity(v_i) = \frac{1}{1 + ||x_{v_i} - x_{CC_i}||}$$

Then, use ranking aggregation to combine metrics considering, more robust to scale differences

$$f_{act}(v_i) = \alpha_0 r_{f_{act}(v_i)} + \alpha_1 r_{CDensity(v_i)}$$

With proper hyper-parameters, we can get a good trade-off between diversity/representativeness and annotation difficulty

Confidence aware prompts + post filtering

We may further use **information generated by LLMs** to filter the selected set of nodes

For example, the confidence of LLMs. If confidence is calibrated, the more confident, the higher the annotation quality

To generate calibrated confidence

1. Let LLMs output TopK results
2. Do k time queries and aggregate results

→ We sort nodes with their confidence and the higher confidence, the higher annotation quality, which shows the effectiveness of our hybrid strategy

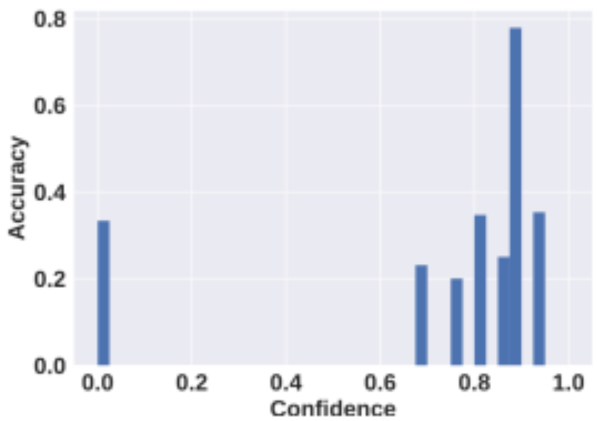
Post filtering

For LLMs' output-based methods

Directly ask LLMs for their confidence

😊**Powerful LLMs may generate high-quality confidence metrics**

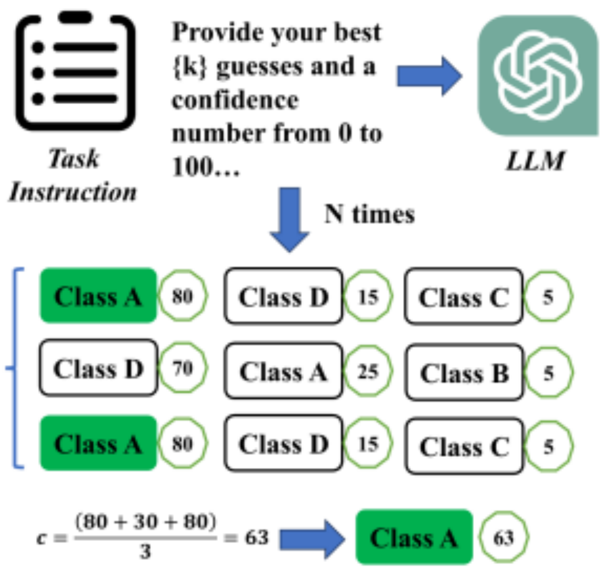
😭**Can only be applied after conducting annotation**



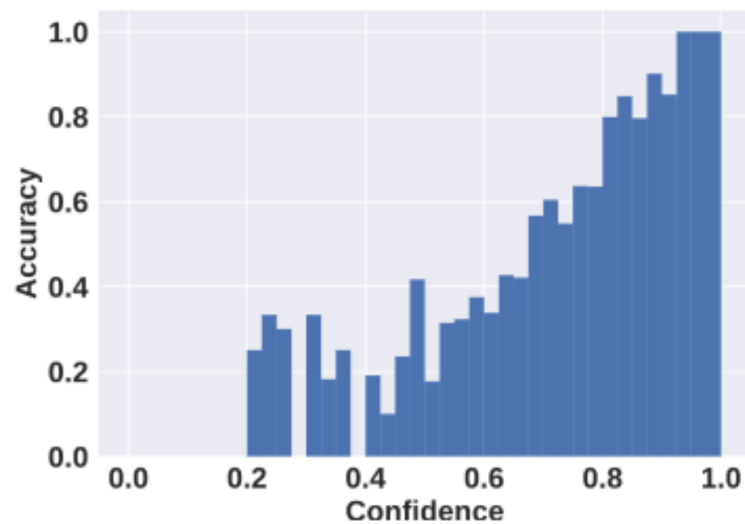
Directly prompt cannot generate accurate and diverse confidence!

Label-free node classification on graphs with LLMs

Confidence Elicitation can help!



Much better!



Two ways to use confidence:

1. Weighted loss for training
2. (optional) filter low-confidence nodes

Table 3: Comparison of label-free node classification methods. The cost is computed in dollars. The performance of methods with * are taken from Li & Hooi (2023). Notably, the time cost of LLMs is proportional to the expenses.

	OGBN-ARXIV		OGBN-PRODUCTS	
Methods	Acc	Cost	Acc	Cost
SES(*)	13.08	N/A	6.67	N/A
TAG-Z(*)	37.08	N/A	47.08	N/A
BART-large-MNLI	13.2	N/A	28.8	N/A
LLMs-as-Predictors	73.33	79	75.33	1572
LLM-GNN	66.32	0.63	74.91	0.74

Good empirical performance and pretty low costs!