LLM Profiling and Fine-Tuning with Limited Neighbor Information for Node Classification on Text-Attributed Graphs

Xinyi Fang
Worcester Polytechnic Institute
Worcester, USA
xfang1@wpi.edu

Kyumin Lee
Worcester Polytechnic Institute
Worcester, USA
kmlee@wpi.edu

Yichuan Li
Worcester Polytechnic Institute
Worcester, USA
yli29@wpi.edu

Abstract*—Graph representation learning has become a critical task across various domains such as social networks and recommender systems. Recently, the rise of large language models (LLMs) has opened up new possibilities for processing text-attributed graphs (TAGs), where nodes are associated with textual information. Despite promising progress, applying LLMs to TAGs faces significant challenges, including input window size limitations and the computational overhead of handling large-scale graphs. To address these challenges, we propose a novel approach that leverages a multi-profiling framework as a data augmentation method, thereby increasing the diversity and quantity of the training samples. The profiles/summaries generated by each of the five profiling/summarizing models are then combined with the graph structure, prompts and ground-truth labels to create a comprehensive and varied fine-tuning data. By strategically selecting profiling models with an appropriate number of neighboring nodes and constructing concise yet informative fine-tuning prompts, our proposed approach enables LLMs to process more complex graphs while operating within limited computational resources. Notably, our experiments demonstrate that it is unnecessary to construct intricate graph structures for fine-tuning to achieve strong performance. Our approach outperforms 11 state-of-the-art baselines, achieving 74.31% accuracy and 85.15% accuracy in two large real-world benchmark datasets, ogbn-arxiv and ogbn-products, respectively. Our code is available at https://github.com/shineef/LLMProfiling.

I. Introduction

In recent years, the development of Large Language Models (LLMs) has revolutionized various domains such as natural language processing and understanding, including their application to text-attributed graphs (TAGs) such as node classification. An example of an arXiv TAG is shown in Figure 1. Each node contains text data, including a title and abstract as its attributes. Each node also has an associated class (e.g., cs.AI: Artificial Intelligence, cs.LG: Machine Learning). Given a node's text attributes with graph structure (e.g., information of its neighbor nodes), the node classification task is to predict its class. Node classification is an important task because of its fundamental usefulness for various domains such as social network analysis and recommender systems.

Despite the success of LLM's applications and importance of the node classification, a couple of significant challenges arise due to 1) the limited context window of the previous methods using LLMs [1], which restricts the amount of information that can be processed in a single prompt and the size of the datasets used in experiments, and 2) high computational requirements of LLMs for processing TAGs. Therefore, although current research on leveraging LLMs for TAGs [2] has proven effective, it is limited to small-scale datasets like PubMed, and Cora. As a result, applying LLMs to larger and more complex TAGs remains a challenge, hindering their practical applicability [3].

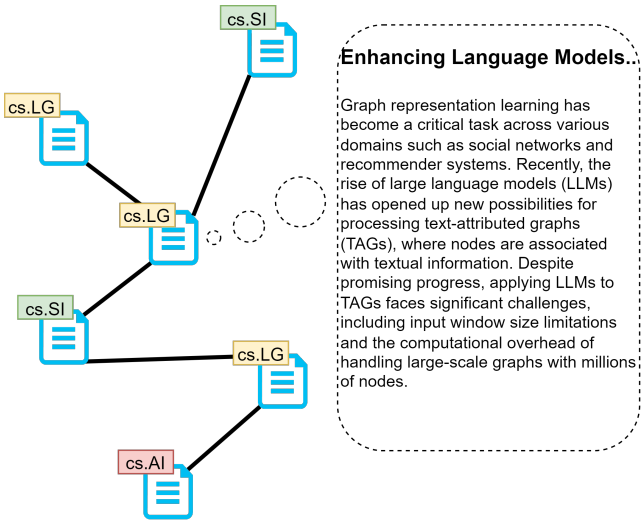

Fig. 1. A simple example of a text-attribute graph.

To address this limitation and enable the effective utilization of LLMs on large graph datasets, we propose an innovative framework which augments training data and takes a limited amount of neighbor information of a target node. Its prompt size for the class prediction is around 1,000 tokens. Our proposed framework consists of three main stages: 1) profile generation as a data augmentation method; 2) fine-tuning data construction; and 3) fine-tuning a backbone LLM. In the first stage, we employ 5 different LLMs to generate 5 different profiles/summaries from a given node's text attributes. The purpose of the profiling method is to leverage the features

of different models to capture the diverse characteristics of the node text toward increasing fine-tuning data size. In the second stage, we combine the original training set with the augmented/generated profiles along with their graph structure information and ground truth labels as part of our designed prompts. In the third stage, we fine-tune an LLM to perform well on the node classification task.

Another key aspects of our research is the exploration of the role of neighbor nodes in both the fine-tuning stage and the inference stage. Graph datasets often exhibit complex relationships and dependencies among nodes. However, pretrained large language models like Llama [4] were already pretrained by using large-scale text data to learn the grammar, contextual relationships, and semantic information of the language. Unlike graph neural networks (GNNs), they already have general inference ability, so they may be able to complete a specific complex task with limited neighbor information (i.e., reduce computational overhead). Our study investigates this aspect and achieves good classification performance with a limited amount of neighbor information of a target node.

In summary, we make the following contributions:

- We propose a novel framework that augments training data by our proposed profiling methods. To the best of our knowledge, we are the first to utilize LLMs to generate profiles for data augmentation in TAGs. The framework fine-tunes an LLM for the node classification in TAGs.
- We investigate the impact of neighbor node information in the fine-tuning process. The result reveals that using a small amount of neighbor node information for fine-tuning can help improve classification accuracy, whereas excessive neighboring information for fine-tuning often leads to reduce the accuracy or requires more computational power.
- The experiment results show that our proposed method outperforms 11 state-of-the-art baselines, achieving 74.31% accuracy and 85.15% accuracy in the ogbn-arxiv and ogbn-products datasets, respectively.

II. Related Work

This section summarizes (1) major machine learning methods for learning node and graph representations for TAGs, (2) data augmentation techniques, and (3) fine-tuning LLMs.

A. Node and Graph Representation Learning Methods for TAGs

Researchers proposed GNN-based methods, integration of LLM with graph-structured data, and transformer-based approaches for learning node and graph representations for TAGs.

In the GNN-based methods, GraphSAGE [5] is an inductive framework for generating node embeddings in previously unseen graphs. GraphSAGE has inspired many follow-up works such as GraphSAINT [6], Cluster-GCN [7] and SimP-GCN [8]. Transformer-based architectures have been adapted to handle TAGs by incorporating the graph structure and textual attributes simultaneously. An example is SPECTER [9]. InstructGLM [1] has proposed that LLMs can replace GNNs.

The integration of LLMs with graph-structured data has emerged as a rapidly growing research direction in recent years [10]. Researchers used LLMs to improve the quality of node embeddings in GNNs, by enhancing node attributes at both the feature level and the text level [11], [12], [13], [14]. Other researchers used LLMs to make predictions on graph-structured data, either by flattening the graph into textual descriptions or by leveraging GNNs to capture structural information to enable LLMs to reason over graph structural information represented in natural language [15], [16], [17], [18]. In some works, GNNs and LLMs were jointly trained to align their embedding spaces, allowing for the mutual enhancement of both modalities [19], [20].

Unlike some of the prior work, our proposed framework is solely based on LLMs to fine-tune and predict a target node's class without using external GNNs.

B. Data Augmentation

Data augmentation is a common technique in machine learning that aims to increase the size and diversity of training datasets. One example is the application of Controlled Graph Translator [21], which aims to transform a given source graph into a target graph while satisfying multiple desired graph attributes at a granular level. Recent advancements in language modeling have inspired researchers to apply language models for data augmentation. *Kumar et al.* [22] investigate different types of Transformer-based pre-trained models for conditional data augmentation. *Chowdhury et al.* [23] propose a two-step generation method to generate context and question-answer pairs. With the development of LLMs, various strategies for data augmentation using LLMs have emerged [24]. These LLM-based data augmentation techniques have been shown to outperform traditional methods [25]. ChatGPT was used as a text data augmentation method [26]. Unlike prior work, we utilize LLMs as profiling methods, which generates diverse profiles/summaries from a given node's textual data.

C. Fine-tuning for Large Models

In recent years, LLMs have achieved remarkable performance across a wide range of NLP tasks. However, to fully harness the power of these models for specific applications, fine-tuning has emerged as a crucial technique [27]. Fine-tuning involves adapting the pre-trained model to a target task using task-specific data or distilling the data generated by the large model into the small model [28]. Several techniques have been developed to enhance the fine-tuning process. Some examples are adapter-based Parameter Efficient Fine-Tuning (PEFT) [29], Low-Rank Adaptation (LoRA), [30], Quantized LoRA (QLoRA) [31], and rank-stabilized LoRA (rsLoRA) [32]. Unsloth has optimized the fine-tuning of large language models by rewriting its kernel using OpenAI's Triton language [33]. In this work, we apply rsLoRA for fine-tuning our model.

III. Our Approach

In this section, we present our proposed framework to address the challenge of applying LLMs to large TAGs. In this paper,

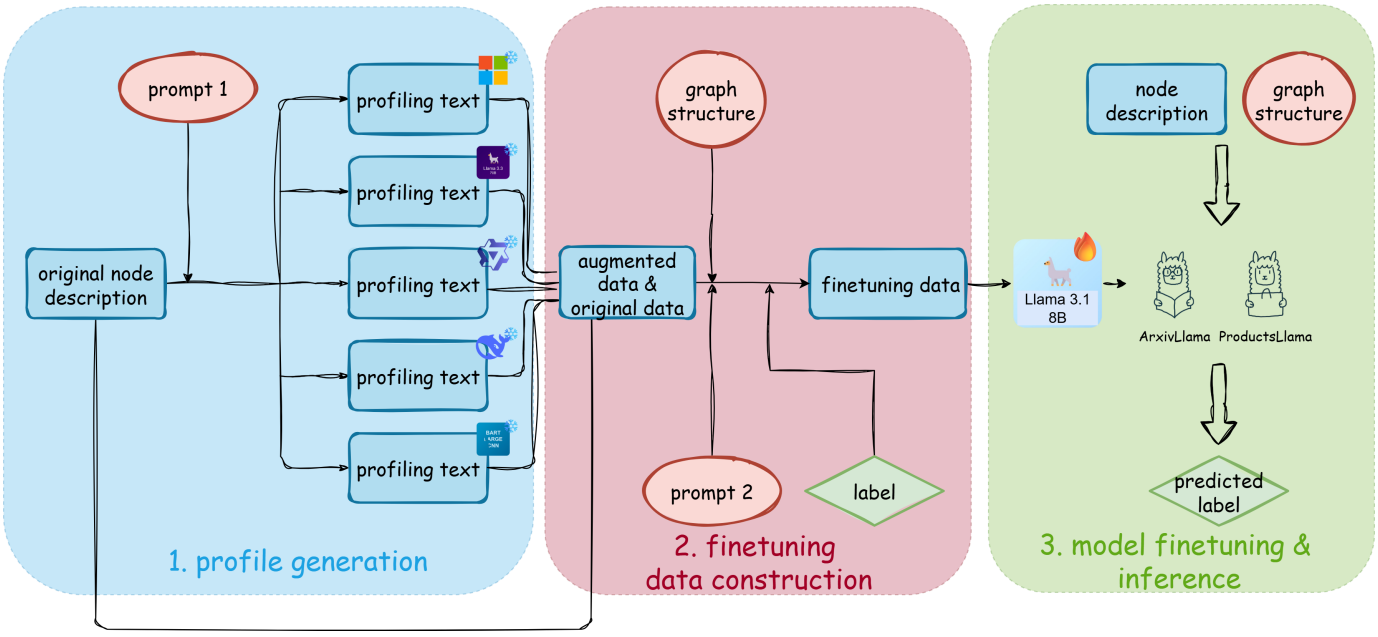

Fig. 2. Our proposed framework.

we use two TAGs: ogbn-arxiv and ogbn-products. Since we construct slightly different prompts for different datasets, we will briefly introduce the datasets in this section. More detailed information about the datasets will be presented in Section IV-A. Our framework consists of three main components: (1) node profile/summary generation as a data augmentation method, (2) fine-tuning data construction, and (3) model fine-tuning and inference. By going through these components/steps, we aim to enable effective learning from the TAGs. The overall framework is shown in Figure 2.

A. Node Profile Generation

We use five LLMs to generate node profiles from the given node text (e.g., paper abstract or product description). Due to the differences among these LLMs in terms of architecture, training data, parameter size, complexity, and training settings, they may have different focuses on capturing the features and characteristics of the same node text. In this case, if LLMs are given appropriate prompts, they will generate profile text with different emphasis. Each generated profile is not comprehensive compared to the original node text, but it is unique in containing certain features or characteristics.

Used LLMs. To generate various profiles/summaries which capture different characteristics of the original node text, we chose five LLMs with different architectures and sizes: Phi-4 14B [34], Llama-3.3 70B [35], QwQ 32B [36], Distilled DeepSeek-R1 based on Qwen2.5 14B [37], and BART [38].

LLM Prompts. In addition to selecting appropriate models for the profile generation, it is also crucial to design a prompt that enables the LLMs to capture relevant features/characteristics from the original node text effectively. If a prompt is overly broad, it will cause each LLM to capture excessive or redundant information and can lead to repetitive and ineffective training or even overfitting during the subsequent fine-tuning process. Conversely, if the prompt is too restrictive and lacks sufficient detail, the resulting absence of essential information can hinder the classifier's ability to learn and classify the data accurately.

We construct a slightly different prompt for each dataset because each dataset has a different dataset name and node text attribute name (e.g., paper abstract in ogbn-arxiv and product description in ogbn-products). Here, we show the prompt for ogbn-arxiv dataset in which the node text data we need to profile is the abstract of each paper. For node v, denote node abstract as A_v, we construct the prompt as follows:

Prompt for Profiling Arxiv Dataset

our task is to summarize the abstract of the paper while retaining as much useful content as possible: A_v
Output the summarized content ONLY.

Next, we show the prompt for ogbn-products in which the node text data we need to profile is the product description of each product. For node v, denote node description as D_v, the constructed prompt is as follows:

Prompt for Profiling Products Dataset

our task is to summarize the description of the product while retaining as much useful content as possible: D_v
Output the summarized content ONLY.

Figure 3 presents five profiles generated from a given node

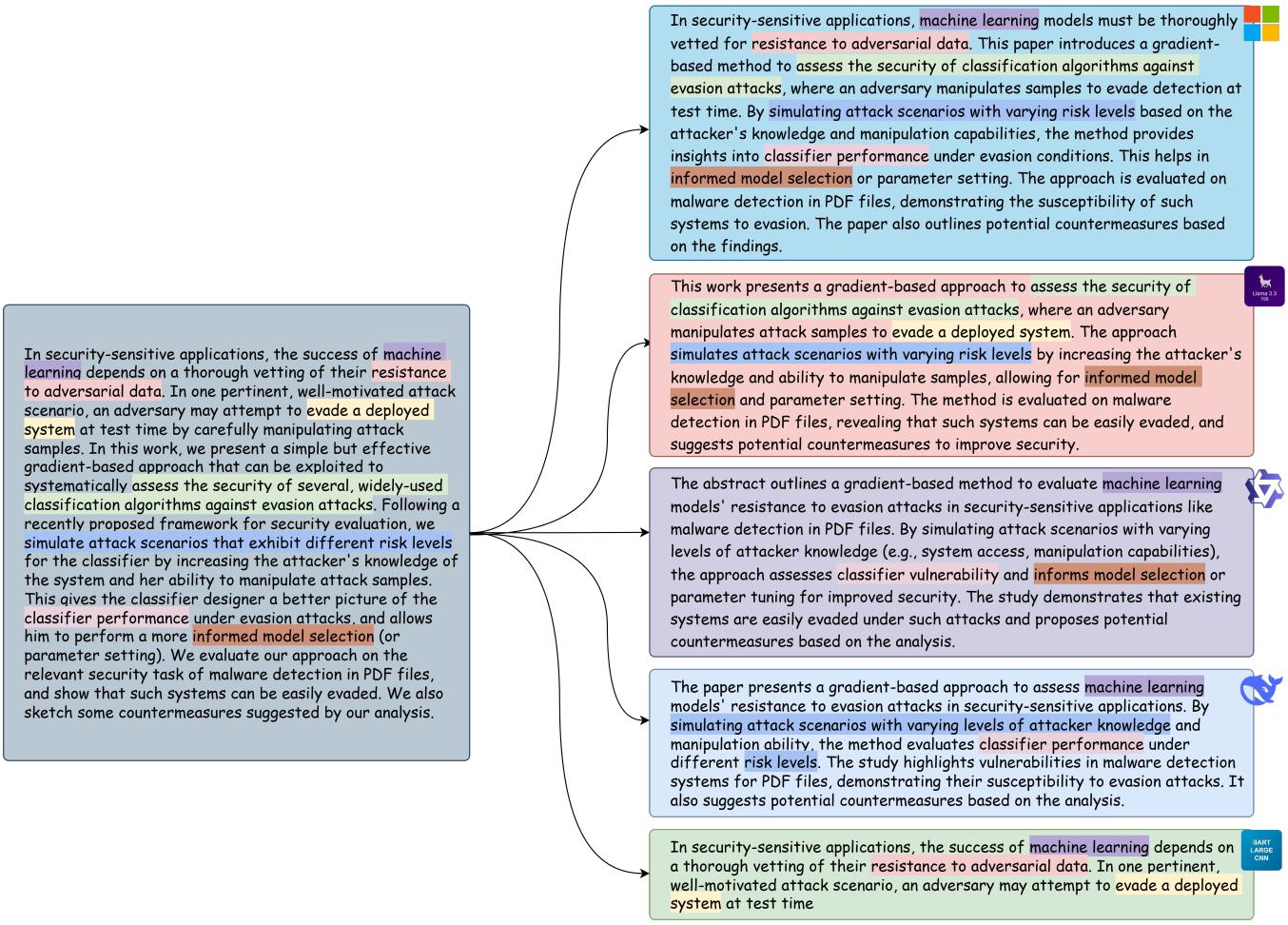

Fig. 3. Original node text and five profiles generated by five LLMs.

text in the ogbn-arxiv dataset as an example. We highlight seven distinct phrases/aspects with different colors in the original node text and also highlight similar phrases with the same colors in the five profiles. As we can see, each profile captures some of the seven phrases (i.e., captures some characteristics/aspects of the original text). When we compare the five generated profiles, their contents are not identical but are relevant to the original node text. It means these generated profiles can be considered as augmented data.

We selected these five LLMs with the expectation that their contributions would differ based on their architectures, sizes and training data. This diversity is critical to our approach, as it ensures that the profiles generated by the LLMs are complementary rather than redundant. Section IV-G shows further in-depth analysis to confirm that this assumption is correct.

B. Fine-tuning Data Construction

In this phase, we construct fine-tuning data which consists of both original data (i.e., training set) and augmented data (i.e., generated profiles). Then, we feed the graph structure and the corresponding true class label with a prompt. Each dataset requires a slightly different prompt because each dataset has a different number of predefined classes, and their names are different.

Here we show ogbn-arxiv dataset's prompt. Let $G = (V, E)$ be a text-attributed graph, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. Each node $v \in V$ has a title T_v and abstract A_v. The edges in E are unweighted and undirected. The prompt using ChatML format for fine-tuning a node classification model is as follows:

Arxiv Prompt for Fine-tuning the Classification Model

```
<|im_start|>system
```

You are a classifier that determines which category a given node belongs to from arxiv cs. The 40 categories are: 0-(Numerical Analysis) 1-(Multimedia) ... 38-

(Digital Libraries) 39-(Discrete Mathematics). Output the corresponding number ONLY

<|im_end|>
<|im_start|>user

Node Title: T_v Abstract: A_v
Neighbors:
- Node Title: T_u (h hop) Abstract: A_u
...

<|im_end|>
<|im_start|>assistant

y_v

<|im_end|>

In the ChatML format, the system prompt contains predefined 40 subject areas of arxiv CS papers. The user prompt contains a central node/target node's information (title and abstract in the Arxiv dataset) and also the graph structure (i.e., selected neighbor nodes' titles and abstracts). Algorithm 1 automatically extracts the given central/target node and its graph structure. In particular, given a central node v, maximum number of hops h_{max} and maximum number of neighbors n_{max} that we are interested in, the breadth-first search algorithm will identify the central node and matched neighbors within the maximum hops and return $D[v]$, a set of the central node and selected neighbor nodes.

Next, we show a prompt for the ogbn-products dataset. In the ogbn-products dataset, let the given graph $G = (V, E)$ be a text-attributed graph and each node $v \in V$ has a product description D_v. The task for this dataset is to predict the category of a product from the 47 top-level categories. It is worth mentioning that the Products dataset does not contain two out of the 47 labels. Therefore, in the system prompt, we only provide information for 45 labels. To extract neighbor nodes along with a central node, we also used the same algorithm presented in Algorithm 1.

After adding system and user prompts in ChatML format with ground truth labels, the example prompt for fine-tuning Llama is as follows:

Product Prompt Example for Fine-tuning

<|im_start|>system

Based on the given product description, determine the most appropriate category. Output only the corresponding category number (no additional text or formatting). Part of the 47 categories are: 0,Home & Kitchen; 1,Health & Personal Care; 2,Beauty; ... 23,Baby Products; 25,Appliances; ... 44,Buy a Kindle; 45,Furniture & Decor

<|im_end|>
<|im_start|>user

Algorithm 1 Neighbor nodes selection along with the central node

procedure CONSTRUCTION(node v, max hops h_{max}, max neighbors n_{max})
  Initialize $D[v] \leftarrow$ central node information
  Initialize an empty set S for visited nodes
  Initialize a queue Q with the node and its initial hop count $(v, 0)$
  Initialize neighbor_count $\leftarrow 0$
  while Q is not empty **and** neighbor_count $< n_{\text{max}}$ **do**
    Remove first element (u, h) from Q
    if u is already visited **or** $h > h_{\text{max}}$ **then**
      continue
    end if
    Mark u as visited by adding it to S
    if $u \neq v$ **then**
      Append neighbor information to $D[v]$: $D[v] \leftarrow h$ hop neighbor information
      Increment neighbor_count
    end if
    for all neighbors w of u **do**
      Add $(w, h + 1)$ to the end of Q
    end for
  end while
end procedure

D_v
Neighbors:
- D_u (h hop) ...

<|im_end|>
<|im_start|>assistant

y_v

<|im_end|>

The user prompt contains the central/target node's product description and selected neighbor nodes' product descriptions.

C. Model Fine-tuning and Inference

The goal of the fine-tuning phase is to build a function $f : V \rightarrow Y$, where $Y = y_1, ..., y_C$ is the set of C classes. Given a node v_i, the function f predicts the label $\hat{y}_i \in Y$. In particular, we adapted a pre-trained large language model, Llama-3.1 8B, as our backbone model for the specific node classification task since it is a relatively lightweight LLM. We use parameter-efficient fine-tuning techniques through the Unsloth framework. The pre-trained model Meta-Llama-3.1-8B-Instruct was loaded in 4-bit quantization mode, which significantly reduced memory usage and allowed efficient fine-tuning even on limited GPU resources. We employed the rsLoRA method for the fine-tuning. The backbone model can be easily replaced with another LLM in practice.

After fine-tuning the Llama with the fine-tuning Arxiv data, we obtained the ArxivLlama model. Similarly, we obtained the

TABLE I
DATASET STATISTICS. THE SPLIT RATIO INDICATES THE PERCENTAGES OF THE TRAINING, VALIDATION, AND TEST SETS.

Dataset	Nodes	Edges	Labels	Evaluation Metric	Split Ratio(%)
ogbn-arxiv	169,343	1,166,243	40	Accuracy	53.7/17.6/28.7
ogbn-products	2,449,029	61,859,140	47	Accuracy	8.03/1.60/90.37

TABLE II
THE AMOUNT OF GENERATED/AUGMENTED PROFILES.

	Phi-4	Llama-3.3	QwQ	DeepSeek-Distill	BART	Total
ogbn-arxiv	91k	30k	45k	10k	91k	**267k**
ogbn-products	90k	81k	14k	24k	197k	**406k**

ProductsLlama model fine-tuned by the fine-tuning Products data. In inference, both ArxivLlama and ProductsLlama take a target node's description with or without its neighbor information as input, and predict its label as output.

IV. EXPERIMENTS

In this section, we conduct a comprehensive evaluation of our proposed approach for the node classification task. We begin by introducing the datasets used in our study and providing details on their characteristics. Then, we present our experiment results and analysis in terms of the effectiveness of our proposed method, the benefit of fine-tuning, hyperparameter tuning, and pairwise similarity between original node texts and generated profiles.

A. Datasets

We utilize two widely used TAG datasets: ogbn-arxiv and ogbn-products. These datasets were chosen because they are highly popular in the research community and are specifically designed for the node classification task. They are relatively large graphs, especially ogbn-products. They come with well-defined tasks and dataset splits, and the original textual data is also accessible.

Table I presents the key statistics of the chosen datasets, including the number of nodes, edges, labels, evaluation metric and split ratio. In the following experiments, we refer to the same split ratio provided on the OGB official website. In particular, the Arxiv dataset is split as follows: train on papers published until 2017, validate on papers published in 2018, and test on papers published since 2019. For the Products dataset, the nodes are split into training, validation, and test sets based on sales rank (popularity). Specifically, the top 8% of products by rank are used for training, the next 2% of products are used for validation, and the remaining products are used for testing.

For each dataset, we will only use the training set for the profile generation. We only use the validation set to evaluate the impact of the profiling approach and the number of neighbor nodes on accuracy (i.e., hyperparameter tuning). The test set is used as our final evaluation of our proposed method. For both datasets, we use 10 random seeds to run the experiments 10 times.

The Arxiv dataset's original training set contains approximately 91k nodes/samples, while the Products dataset's original training set consists of around 197k nodes. The number of generated/augmented profiles for Arxiv and Products datasets are 267k and 406k, respectively. Table II shows a detailed number of generated profiles per LLM. We used both training set and generated profiles for constructing fine-tuning data as we mentioned in the previous section.

B. Baselines and Experiment Setting

We consider 11 baselines for the node classification task:

- **MLP** [39]: A multilayer perceptron predictor that uses the given raw node features.
- **GraphSAGE** [5]: A popular GNN model that learns node embeddings through sampling and aggregating neighborhood information.
- **GCN** [40]: A GNN model that learns node representations by convolving the graph structure with node features.
- **E2EG** [41]: A transformer-based model designed for text-attributed graphs.
- **LLM4NG** [2]: It utilizes LLMs to extract semantic information from the labels and generate samples that belong to these categories/labels as exemplars.
- **CoarFormer** [42]: A transformer-based model that combines coarse-grained and fine-grained attention to capture both global and local information in graphs.
- **GOAT** [43]: A graph-based pre-training framework that leverages the power of language models to learn graph representations.
- **LargeGT** [44]: A large-scale graph transformer model that creates a fast neighborhood sampling technique coupled with a local attention mechanism to scale up graph transformer architectures to handle larger graphs.
- **Polynormer** [45]: It is a polynomial-expressive graph transformer model with linear complexity, and learns a high-degree polynomial on input features.
- **AskGNN** [46]: It leverages In-Context Learning to integrate graph data and task-specific information into LLMs. AskGNN adopts Qwen1.5-72B and Llama3-70B as its primary inference LLMs. Since Llama3-70B achieves better results, we included it as a baseline in our comparisons.

TABLE III
TRAINING PARAMETERS AND LORA CONFIGURATION.

Parameter	Value
Learning Rate	3e-4
Batch Size	8
Gradient Accumulation Steps	2
Number of Epochs	1
Logging Steps	1
Optimizer	AdamW
Weight Decay	0.01
Warmup Steps	10
Random Seed	0
LoRA Rank	16
LoRA Alpha	16

TABLE IV
HARDWARE SPECIFICATIONS FOR FINE-TUNING.

Component	Specification
GPU	NVIDIA RTX A5000 (24GB memory)
Operating System	Ubuntu 20.04
Deep Learning Framework	PyTorch 2.5
CUDA Version	CUDA 12.2
Python Version	Python 3.11

- **LLM-GNN** [47]: It adopts gpt-3.5-turbo to generate high-quality annotations for a subset of graph nodes, which are then used to train a GNN for predicting the remaining nodes.

Performance of the baselines is directly obtained from the authors' published papers since the authors thoroughly optimized their own models based on the same benchmark datasets with the predefined same split ratios, which are consistent with our setting. Following prior work [48], we chose learning rate = 0.0003, batch size = 8 and number of epochs = 1 in the fine-tuning phase for our models. We outline the training parameters and LoRA configuration used in our experiments in Table III.

The hardware specifications used for fine-tuning our model are presented in Table IV. Our models were fine-tuned on an Nvidia RTX A5000 GPU, a relatively inexpensive resource, because of our lightweight backbone model selection and limited neighbor information requirement. The training took only 67 hours (less than 3 days) by solely using the A5000 GPU. If we use multiple GPUs and/or multiple machines, the training time will be further reduced. Following the research community in this domain, we use accuracy as the evaluation metric.

C. Our approach vs. Baselines

To evaluate the effectiveness of our fine-tuned models, ArxivLlama and ProductsLlama, we compare their performance against the nine baselines. In the inference, our models use a target node's text information along with information of 5 neighbor nodes in 1 hop (i.e., limited graph structure information).

Table V presents the comparative results. In both datasets, our proposed method outperformed all the baselines, achieving 74.31% and 85.15% on the ogbn-arxiv and ogbn-products, respectively. The experimental result shows the effectiveness of our models.

TABLE V
PERFORMANCE OF OUR MODELS VS. BASELINES.

Model	ogbn-arxiv Accuracy (%)	ogbn-products Accuracy (%)
MLP	55.50 ± 0.23	61.06 ± 0.08
GraphSAGE	72.95 ± 0.31	83.89 ± 0.36
GCN	73.60 ± 0.18	82.33 ± 0.19
E2EG	73.62 ± 0.14	80.98 ± 0.40
LLM4NG	61.40 ± 1.06	-
CoarFormer	71.66 ± 0.24	79.18 ± 0.20
GOAT	72.41 ± 0.40	82.00 ± 0.43
LargeGT	-	79.81 ± 0.25
Polynormer	73.46 ± 0.16	83.82 ± 0.11
AskGNN	71.53 ± 0.11	82.88 ± 0.17
LLM-GNN	66.32 ± 0.15	74.91 ± 0.17
ArxivLlama	$\mathbf{74.31 \pm 0.08}$	-
ProductsLlama	-	$\mathbf{85.15 \pm 0.01}$

TABLE VI
PERFORMANCE OF ZERO-SHOT PROMPTING, FEW-SHOT PROMPTING AND FINE-TUNED MODEL IN THE VALIDATION SET.

Zero-shot	3-shot	Fine-tuning
25.01%	26.18%	74.57%

D. Zero-shot, Few-shot and Fine-tuning

In this experiment, we compare zero-shot and few-shot prompting with fine-tuning to justify why fine-tuning is necessary for the node classification task. We use Llama-3.1 8B as the backbone model.

In the zero-shot prompting, we only feed a node's information (e.g., title and abstract of the node in the Arxiv dataset) without its neighbor information, and ask Llama to predict the node's class label. In the few-shot prompting, we additional feed few examples and their true labels to the Llama. In this fine-tuning setting, we only use the original training set without augmented data so that we can measure the pure effectiveness of fine-tuning without data augmentation. During the fine-tuning, we only feed each training example's node information without its neighbor information. In the inference, we also feed a target node's information only without its neighbor information.

Table VI presents the prediction results on the validation set of the Arxiv dataset. The fine-tuned model achieves 74.57% accuracy with a significant improvement compared with zero-shot and few-shot prompting. The results confirm the effectiveness of the fine-tuning for the node classification task.

E. Varying Fine-tuning Data and Number of Neighbors

In the previous experiment, we learned that fine-tuning is very effective. In this section, we dive deeper into the fine-tuning setting, especially varying fine-tuning data in the fine-tuning stage and the number of neighbors in the inference stage as hyperparameters. In this experiment, we use the original

training set with and/or without augmented data for fine-tuning, and evaluate the performance of the models on the validation set of the Arxiv dataset with tuning the number of neighbors in the node description in the inference stage.

First of all, we compare two models with different fine-tuning data: 1) a model fine-tuned by the original training set without neighbor information; and 2) another model fine-tuned by the original training set with neighbor information. Figure 4 shows experimental results. The model fine-tuned by the original training set with 5 neighbor information in 1 hop (yellow line) achieved more consistent results than the other model using the original training set without neighbor information (orange line) when we varied the number of neighbors as part of node description in the inference stage (from 0 to 40 neighbor nodes' information) although latter one achieved better results on 0 neighbor information in the inference stage.

Next research question is what if we combine both original training set *with* 5 neighbor information and the same original training set *without* the neighbor information? Can we achieve better and consistent results since one of them achieved consistent results and the other one achieved better results with 0 neighbor information in the inference stage? Yes, when we look at the red line in the figure, a model fine-tuned by the combined data achieved consistent and better performance, making a balance between them.

The next research question is can we further improve the performance of our model using all the data (i.e., original training set with neighbor information, the same set without neighbor information, and generated/augmented profiles) for fine-tuning? The blue line is a model fine-tuned by all the data including profiles generated by five LLMs. Overall, the blue line achieved the best performance in 5 neighbors and 25 neighbors within 1 hop in the inference stage on the validation set, indicating all types of data contributed positively in the fine-tuning stage. In our final model, we chose 5 neighbors in the inference on the test set to reduce the prompt size. The experimental results confirm that our proposed framework works well and is effective.

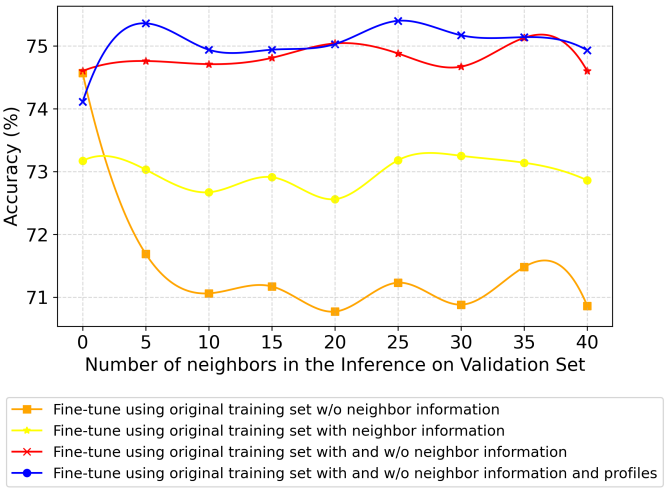

Fig. 4. Performance of our models by varying fine-tuning data and number of neighbors in the inference on the validation set.

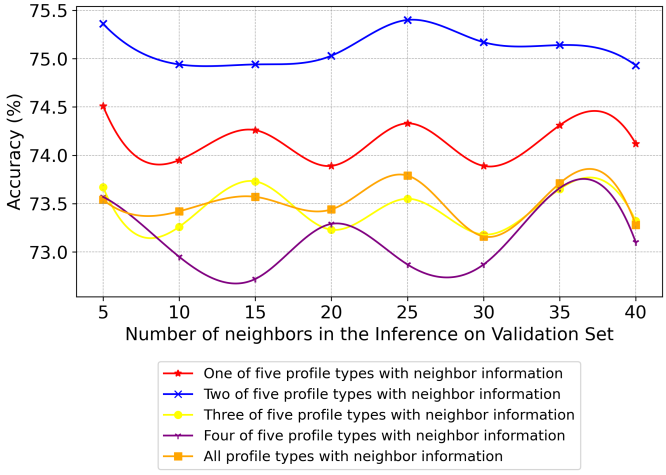

Fig. 5. Performance of our models by varying number of profile types with neighbor information for fine-tuning and number of neighbors in the inference on the validation set.

F. Varying Number of Profile Types with Neighbor Information

When we construct fine-tuning prompts, we can choose whether we will add graph structure (i.e., neighbor information) or not along with a target/central node's text information (e.g., paper title and abstract) into the user prompt. In this experiment, we vary how many generated profile types will come with neighbor information. Since we have five profile types generated by five LLMs, we start from 1 profile type with neighbor information and increase until all profiling types with neighbor information. Note that we always keep the original training set with 5 neighbors in 1 hop in the experiment as we found that this is a good setting from the previous section.

Figure 5 presents the experiment results in the Arxiv dataset. Among the five variants of our models, two profile types (generated by Llama and DeepSeek) with neighbor information and remaining three profile types without neighbor information achieved the best results. Therefore, our final models used this setting in Section IV-C.

G. Similarity between Original Node Text and Generated Profile

To quantify the complementary "focus" contributed by each LLM, we measured cosine similarity between each original node text and its corresponding profile generated by by each LLM model. Figure 6 shows the mean cosine similarity score and standard deviation of each LLM. The upper panel displays a scatter plot comparing the mean similarity scores against their standard deviations for all five LLMs on the two datasets. Blue points represent profiling models evaluated on the ogbn-arxiv dataset, while orange points represent ones on the ogbn-products dataset. Each point's position reflects a

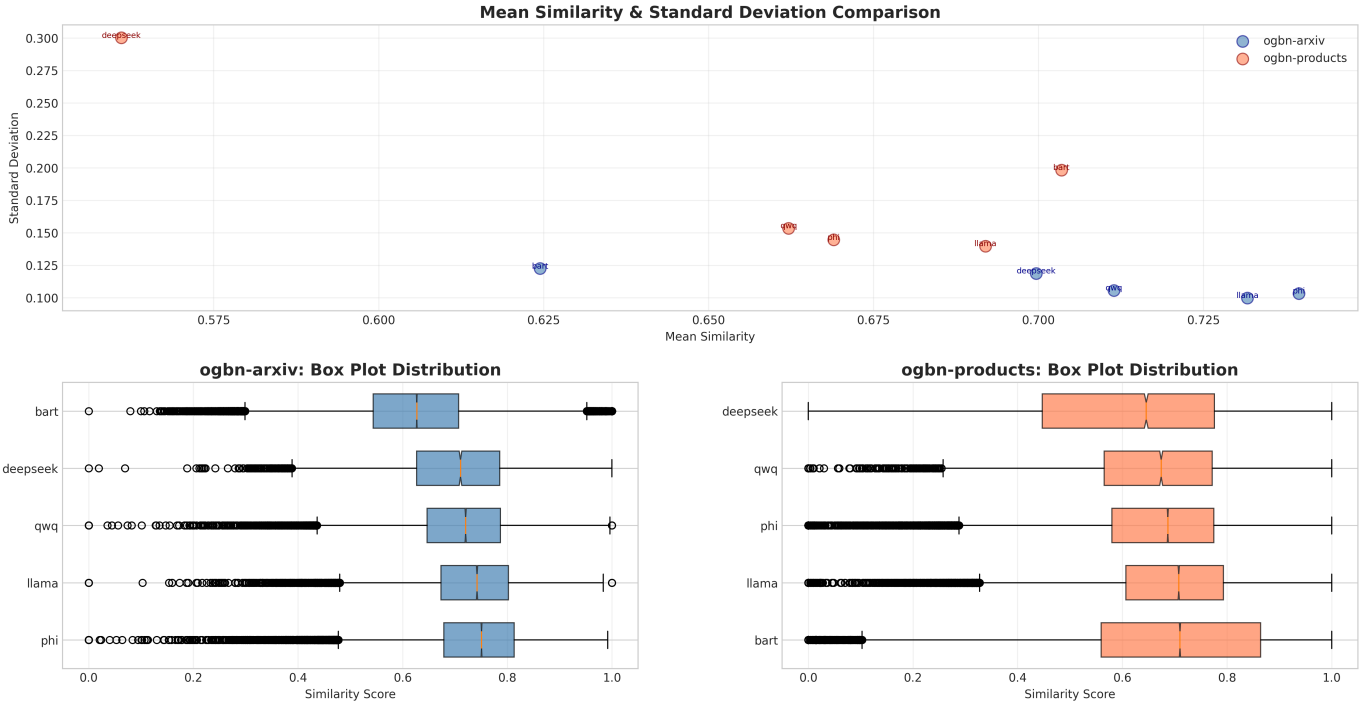

Fig. 6. The mean similarity and standard deviation analysis on the ogbn-arxiv and ogbn-products dataset.

profiling model's mean cosine similarity score between original node texts and generated profiles on the x-axis, and standard deviation of their cosine similarity scores on the y-axis. It can be observed that each model exhibits a distinct ability to generate profiles through their similarity scores. In particular, in the Arxiv dataset, the BART model demonstrates the lowest mean similarity while in the Products dataset, the DeepSeek model shows notably high variance with a lower mean similarity, whereas the BART model achieves the highest mean similarity.

The lower panels present horizontal box plots from each dataset, ordered by the mean cosine similarity score. The Arxiv dataset exhibits relatively more compact distributions compared to the Products dataset. But for both datasets, BART and DeepSeek display greater variability with more widespread distributions, whereas QwQ, Llama, and Phi demonstrate more concentrated distributions. This analysis confirms that the non-overlapping notches among the profiling models indicate statistically significant differences, and these profiling/summary differences contributed to improved performance.

V. Discussion and Limitations

One primary limitation of our method is that we did not exhaustively explore all possible LLM hyperparameters such as learning rate. However, our model still surpassed the performance of the state-of-the-art models. This suggests that there is potential for further improving our model's performance.

Secondly, we did not explore automatically selecting optimal number of profiling methods. An interesting research question is to build another model, which automatically select the best combination of profiling methods. For example, among 100 LLM profiling models, which ones will be used and how many models will lead to achieve the best results. We leave this research in the future work.

Lastly, we only employed a light Llama-3.1 8B as our backbone model. In fact, more complex and larger models such as Llama-3.1 70B, may produce even better performance. However, our light backbone model selection allows our model to be fine-tuned on a single A5000 GPU, significantly reducing the GPU resource requirements and consumption.

VI. Conclusion

In this paper, we proposed an LLM-based novel framework, which generated profiles from the original training set to increase the size of fine-tuning data, carefully designed the fine-tuning data, and then fine-tuned Llama-3.1 8B as a backbone model. In hyperparameter tuning, we thoroughly compared variants of the fine-tuning data and each node's neighbor information. In the inference, we also thoroughly chose the best neighbor information. Our proposed models with limited neighbor information outperformed nine state-of-the-art baselines on two large text-attributed graphs.

Acknowledgments

This work was supported in part by NSF grant IOS-2430277.

References

[1] R. Ye, C. Zhang, R. Wang, S. Xu, and Y. Zhang, "Language is all a graph needs," *EACL*, 2024.

[2] J. Yu, Y. Ren, C. Gong, J. Tan, X. Li, and X. Zhang, "Leveraging large language models for node generation in few-shot learning on text-attributed graphs," in *AAAI*, 2025.

[3] A. Zolnai-Lucas, J. Boylan, C. Hokamp, and P. Ghaffari, "STAGE: Simplified text-attributed graph embeddings using pre-trained LLMs," in *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, 2024.

[4] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

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

[6] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Accurate, efficient and scalable graph embedding," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2019.

[7] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *KDD*, 2019.

[8] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *WSDM*, 2021.

[9] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. S. Weld, "SPECTER: Document-level Representation Learning using Citation-informed Transformers," in *ACL*, 2020.

[10] Y. Li, Z. Li, P. Wang, J. Li, X. Sun, H. Cheng, and J. X. Yu, "A survey of graph meets large language model: progress and future directions," in *IJCAI*, 2024.

[11] 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. Newsl.*, vol. 25, no. 2, p. 42–61, Mar. 2024.

[12] E. Chien, W.-C. Chang, C.-J. Hsieh, H.-F. Yu, J. Zhang, O. Milenkovic, and I. S. Dhillon, "Node feature extraction by self-supervised multi-scale neighborhood prediction," in *ICLR*, 2022.

[13] 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 *ICLR*, 2024.

[14] B. Pan, Z. Zhang, Y. Zhang, Y. Hu, and L. Zhao, "Distilling large language models for text-attributed graph learning," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 1836–1845.

[15] Y. Hu, Z. Zhang, and L. Zhao, "Beyond text: A deep dive into large language models' ability on understanding graph data," in *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*.

[16] B. Perozzi, B. Fatemi, D. Zelle, A. Tsitsulin, M. Kazemi, R. Al-Rfou, and J. Halcrow, "Let your graph do the talking: Encoding structured data for llms," 2024. [Online]. Available: https://arxiv.org/abs/2402.05862

[17] B. Fatemi, J. Halcrow, and B. Perozzi, "Talk like a graph: Encoding graphs for large language models," in *International Conference on Learning Representations (ICLR)*, 2024.

[18] D. Das, I. Gupta, J. Srivastava, and D. Kang, "Which modality should I use - text, motif, or image? : Understanding graphs with large language models," in *NAACL*, 2024.

[19] Y. Li, K. Ding, and K. Lee, "GRENADE: Graph-centric language model for self-supervised representation learning on text-attributed graphs," in *EMNLP*, 2023.

[20] B. Jin, W. Zhang, Y. Zhang, Y. Meng, X. Zhang, Q. Zhu, and J. Han, "Patton: Language model pretraining on text-rich networks," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

[21] N. Vakil and H. Amiri, "Controlled transformation of text-attributed graphs," in *EMNLP*, 2024.

[22] V. Kumar, A. Choudhary, and E. Cho, "Data augmentation using pre-trained transformer models," in *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, 2020, pp. 18–26.

[23] A. Chowdhury and A. Chadha, "Generative data augmentation using LLMs improves distributional robustness in question answering," in *EACL: Student Research Workshop*, 2024.

[24] B. Ding, C. Qin, R. Zhao, T. Luo, X. Li, G. Chen, W. Xia, J. Hu, A. T. Luu, and S. Joty, "Data augmentation using LLMs: Data perspectives, learning paradigms and challenges," in *ACL Findings*, 2024.

[25] Y. Zhou, C. Guo, X. Wang, Y. Chang, and Y. Wu, "A survey on data augmentation in large model era," *arXiv preprint arXiv:2401.15422*, 2024.

[26] H. Dai, Z. Liu, W. Liao, X. Huang, Y. Cao, Z. Wu, L. Zhao, S. Xu, F. Zeng, W. Liu *et al.*, "Auggpt: Leveraging chatgpt for text data augmentation," *IEEE Transactions on Big Data*, 2025.

[27] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, "Scaling instruction-finetuned language models," *Journal of Machine Learning Research*, vol. 25, no. 70, pp. 1–53, 2024.

[28] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[29] Z. Hu, L. Wang, Y. Lan, W. Xu, E.-P. Lim, L. Bing, X. Xu, S. Poria, and R. Lee, "LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[30] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[31] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *Advances in neural information processing systems*, vol. 36, pp. 10 088–10 115, 2023.

[32] D. Kalajdzievski, "A rank stabilization scaling factor for fine-tuning with lora," 2023. [Online]. Available: https://arxiv.org/abs/2312.03732

[33] M. H. Daniel Han and U. team, "Unsloth," 2023. [Online]. Available: http://github.com/unslothai/unsloth

[34] M. Abdin, J. Aneja, H. Behl, S. Bubeck, R. Eldan, S. Gunasekar, M. Harrison, R. J. Hewett, M. Javaheripi, P. Kauffmann *et al.*, "Phi-4 technical report," *arXiv preprint arXiv:2412.08905*, 2024.

[35] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[36] Q. Team, "Qwq-32b: Embracing the power of reinforcement learning," March 2025. [Online]. Available: https://qwenlm.github.io/blog/qwq-32b/

[37] DeepSeek-AI, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[38] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[39] 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," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

[40] Y. Luo, L. Shi, and X.-M. Wu, "Classic GNNs are strong baselines: Reassessing GNNs for node classification," in *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. [Online]. Available: https://openreview.net/forum?id=xkljKdGe4E

[41] T. A. Dinh, J. den Boef, J. Cornelisse, and P. Groth, "E2eg: End-to-end node classification using graph topology and text-based node attributes," in *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2023, pp. 1084–1091.

[42] W. Kuang, Z. WANG, Y. Li, Z. Wei, and B. Ding, "Coarformer: Transformer for large graph via graph coarsening," 2022. [Online]. Available: https://openreview.net/forum?id=fkjO_FKVzw

[43] K. Kong, J. Chen, J. Kirchenbauer, R. Ni, C. B. Bruss, and T. Goldstein, "Goat: A global transformer on large-scale graphs," in *International Conference on Machine Learning*. PMLR, 2023, pp. 17 375–17 390.

[44] V. P. Dwivedi, Y. Liu, A. T. Luu, X. Bresson, N. Shah, and T. Zhao, "Graph transformers for large graphs," *arXiv preprint arXiv:2312.11109*, 2023.

[45] C. Deng, Z. Yue, and Z. Zhang, "Polynormer: Polynomial-expressive graph transformer in linear time," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=hmv1LpNfXa

[46] Z. Hu, Y. Li, Z. Chen, J. Wang, H. Liu, K. Lee, and K. Ding, "Let's ask gnn: Empowering large language model for graph in-context learning," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 1396–1409.

[47] 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)," *arXiv preprint arXiv:2310.04668*, 2023.

[48] M. Labonne, "Fine-tune llama 3.1 ultra-efficiently with unsloth," 2024. [Online]. Available: https://huggingface.co/blog/mlabonne/sft-llama3