Exploring the Potential of Large Language Models (LLMs) in Learning on Graphs

Zhikai Chen[1], Haitao Mao[1], Hang Li[1], Wei Jin[3], Hongzhi Wen[1],
Xiaochi Wei[2], Shuaiqiang Wang[2], Dawei Yin[2]
Wenqi Fan[4], Hui Liu[1], Jiliang Tang[1]
[1]Michigan State University [2] Baidu Inc. [3] Emory University
[4] The Hong Kong Polytechnic University
{chenzh85, haitaoma, lihang4, wenhongz, liuhui7, tangjili}@msu.edu,
{weixiaochi, wangshuaiqiang}@baidu.com, yindawei@acm.org,
wei.jin@emory.edu,
wenqifan03@gmail.com

ABSTRACT

Learning on Graphs has attracted immense attention due to its wide real-world applications. The most popular pipeline for learning on graphs with textual node attributes primarily relies on Graph Neural Networks (GNNs), and utilizes shallow text embedding as initial node representations, which has limitations in general knowledge and profound semantic understanding. In recent years, Large Language Models (LLMs) have been proven to possess extensive common knowledge and powerful semantic comprehension abilities that have revolutionized existing workflows to handle text data. In this paper, we aim to explore the potential of LLMs in graph machine learning, especially the node classification task, and investigate two possible pipelines: *LLMs-as-Enhancers* and *LLMs-as-Predictors*. The former leverages LLMs to enhance nodes' text attributes with their massive knowledge and then generate predictions through GNNs. The latter attempts to directly employ LLMs as standalone predictors. We conduct comprehensive and systematical studies on these two pipelines under various settings. From comprehensive empirical results, we make original observations and find new insights that open new possibilities and suggest promising directions to leverage LLMs for learning on graphs. Our codes and datasets are available at: `https://github.com/CurryTang/Graph-LLM` .

1. INTRODUCTION

Graphs are ubiquitous in various disciplines and applications, encompassing a wide range of real-world scenarios [73]. Many of these graphs have nodes that are associated with text attributes, resulting in the emergence of text-attributed graphs, such as citation graphs [23; 57] and product graphs [5]. For example, in the OGBN-PRODUCTS dataset [23], each node represents a product, and its corresponding textual description is treated as the node's attribute. These graphs have seen widespread use across a myriad of domains, from social network analysis [31], information retrieval [86], to a diverse range of natural language processing tasks [37; 76].

Given the prevalence of text-attributed graphs (TAGs), we aim to explore how to effectively handle these graphs, with a focus on the node classification task. Intuitively, TAGs provide both node attribute and graph structural information. Thus, it is important to effectively capture both while modeling their interrelated correlation. Graph Neural Networks (GNNs) [38] have emerged as the de facto technique for handling graph-structured data, often leveraging a message-passing paradigm to effectively capture the graph structure. To encode textual information, conventional pipelines typically make use of non-contextualized shallow embeddings e.g., Bag-of-Words [20] and Word2Vec [42] embeddings, as seen in the common graph benchmark datasets [23; 57], where GNNs are subsequently employed to process these embeddings. Recent studies demonstrate that these non-contextualized shallow embeddings suffer from some limitations, such as the inability to capture polysemous words [51] and deficiency in semantic information [41; 12], which may lead to sub-optimal performance on downstream tasks.

Compared to these non-contextualized shallow textual embeddings, large language models (LLMs) present massive context-aware knowledge and superior semantic comprehension capability through the process of pre-training on large-scale text corpora [48; 12]. This knowledge achieved from pre-training has led to a surge of revolutions for downstream NLP tasks [85]. Exemplars such as ChatGPT and GPT4 [46], equipped with hundreds of billions of parameters, exhibit superior performance [2] on numerous text-related tasks from various domains. Considering the exceptional ability of these LLMs to process and understand textual data, a pertinent question arises: (1) *Can we leverage the knowledge of LLMs to compensate for the deficiency of contextualized knowledge and semantic comprehension inherent in the conventional GNN pipelines?* In addition to the knowledge learned via pre-training, recent studies suggest that LLMs present preliminary success on tasks with implicit graph structures such as recommendation [35; 14], ranking [26], and multi-hop reasoning [7], in which LLMs are adopted to make the final predictions. Given such success, we further question: (2) *Can LLMs, beyond merely integrating with GNNs, independently perform predictive tasks with explicit graph structures?* In this paper, we aim to embark upon a preliminary investigation of these two questions by undertaking a series of extensive empirical analyses. Particularly, the key challenge is how to design an LLM-compatible pipeline for graph learn-

ing tasks. Consequently, we explore two potential pipelines to incorporate LLMs: (1) *LLMs-as-Enhancers*: LLMs are adopted to enhance the textual information; subsequently, GNNs utilize refined textual data to generate predictions. (2) *LLMs-as-Predictors*: LLMs are adapted to generate the final predictions, where structural and attribute information is present completely through natural languages.

In this work, we embrace the challenges and opportunities to study the utilization of LLMs in graph-related problems and aim to deepen our understanding of *the potential of LLMs on graph machine learning*, with a focus on the node classification task. **First**, we aim to investigate how LLMs can enhance GNNs by leveraging their extensive knowledge and semantic comprehension capability. It is evident that different types of LLMs possess varying levels of capability, and more powerful models often come with more usage restrictions [59; 85; 51]. Therefore, we strive to design different strategies tailored to different types of models, and better leverage their capabilities within the constraints of these usage limitations. **Second**, we want to explore how LLMs can be adapted to explicit graph structures as a predictor. A principal challenge lies in crafting a prompt that enables the LLMs to effectively use structural and attribute information. To address this challenge, we attempt to explore what information can assist LLMs in better understanding and utilizing graph structures. Through these investigations, we make some insightful observations and gain a better understanding of the capabilities of LLMs in graph machine learning.

Contributions. Our contributions are summarized as follows:

1. We explore two pipelines that incorporate LLMs to handle TAGs: *LLMs-as-Enhancers* and *LLMs-as-Predictors*. The first pipeline treats the LLMs as attribute enhancers, seamlessly integrating them with GNNs. The second pipeline directly employs the LLMs to generate predictions.
2. For *LLMs-as-Enhancers*, we introduce two strategies to enhance text attributes via LLMs. We further conduct a series of experiments to compare the effectiveness of these enhancements.
3. For *LLMs-as-Predictors*, we design a series of experiments to explore LLMs' capability in utilizing structural and attribute information. From empirical results, we summarize some original observations and provide new insights.

Key Insights. Through comprehensive empirical evaluations, we find the following key insights:

1. For *LLMs-as-Enhancers*, using deep sentence embedding models to generate embeddings for node attributes show both effectiveness and efficiency.
2. For *LLMs-as-Enhancers*, utilizing LLMs to augment node attributes at the text level also leads to improvements in downstream performance.
3. For *LLMs-as-Predictors*, LLMs present preliminary effectiveness but we should be careful about their inaccurate predictions and the potential test data leakage problem.
4. LLMs demonstrate the potential to serve as good annotators for labeling nodes, as a decent portion of their annotations is accurate.

Organization. The remaining of this paper is organized as follows. Section 2 introduces necessary preliminary knowledge and notations used in this paper. Section 3 introduces two pipelines to leverage LLMs under the task of node classification. Section 4 explores the first pipeline, *LLMs-as-Enhancers*, which adopts LLMs to enhance text attributes. Section 5 details the second pipeline, *LLMs-as-Predictors*, exploring the potential for directly applying LLMs to solve graph learning problems as a predictor. Section 6 discusses works relevant to the applications of LLMs in the graph domain. Section 7 summarizes our insights and discusses the limitations of our study and the potential directions of LLMs in the graph domain.

2. PRELIMINARIES

In this section, we present concepts, notations and problem settings used in the work. We primarily delve into the node classification task on the text-attributed graphs, which is one of the most important downstream tasks in the graph learning domain. Next, we first give the definition of text-attributed graphs.

Text-Attributed Graphs A text-attributed graph (TAG) $\mathcal{G}_S$ is defined as a structure consisting of nodes $\mathcal{V}$ and their corresponding adjacency matrix $\mathbf{A} \in \mathbb{R}^{|V|\times|V|}$. For each node $v_i \in \mathcal{V}$, it is associated with a text attribute, denoted as $\mathbf{s}_i$.

In this study, we focus on node classification, which is one of the most commonly adopted graph-related tasks.

Node Classification on TAGs Given a set of labeled nodes $\mathcal{L} \subset \mathcal{V}$ with their labels $y_{\mathcal{L}}$, we aim to predict the labels $\mathbf{y}_{\mathcal{U}}$ for the remaining unlabeled nodes $\mathcal{U} = \mathcal{V} \setminus \mathcal{L}$.

We use the citation network dataset Ogbn-arxiv [23] as an illustrative example. In such a graph, each node represents an individual paper from the computer science subcategory, with the attribute of the node embodying the paper's title and abstracts. The edges denote the citation relationships. The task is to classify the papers into their corresponding categories, for example, "cs.cv" (i.e., computer vision). Next, we introduce the models adopted in this study, including graph neural networks and large language models.

Graph Neural Networks. When applied to TAGs for node classification, Graph Neural Networks (GNNs) leverage the structural interactions between nodes. Given initial node features h_i^0, GNNs update the representation of each node by aggregating the information from neighboring nodes in a message-passing manner [16]. The l-th layer can be formulated as:

$$h_i^l = \mathrm{UPD}^l\left(h_i^{l-1}, \mathrm{AGG}_{j\in\mathcal{N}(i)}\,\mathrm{MSG}^l\left(h_i^{l-1}, h_j^{l-1}\right)\right), \quad (1)$$

where AGG is often an aggregation function such as summation, or maximum. UPD and MSG are usually some differentiable functions, such as MLP. The final hidden representations can be passed through a fully connected layer to make classification predictions.

Large Language Models. In this work, we primarily utilize the term "large language models" (LLMs) to denote language models that have been pre-trained on extensive text corpora. Despite the diversity of pre-training objectives [9; 52; 53], the shared goal of these LLMs is to harness the knowledge acquired during the pre-training phase and repurpose it for a range of downstream tasks. Based on their interfaces, specifically considering whether their embeddings are accessible to users or not, in this work we roughly classify LLMs as below:

Embedding-visible LLMs Embedding-visible LLMs pro-

vide access to their embeddings, allowing users to interact with and manipulate the underlying language representations. Embedding-visible LLMs enable users to extract embeddings for specific words, sentences, or documents, and perform various natural language processing tasks using those embeddings. Examples of embedding-visible LLMs include BERT [9], Sentence-BERT [54], and Deberta [21].

Embedding-invisible LLMs Embedding-invisible LLMs do not provide direct access to their embeddings or allow users to manipulate the underlying language representations. Instead, they are typically deployed as web services [59] and offer restricted interfaces. For instance, ChatGPT [45], along with its API, solely provides a text-based interface. Users can only engage with these LLMs through text interactions.

In addition to the interfaces, the size, capability, and model structure are crucial factors in determining how LLMs can be leveraged for graphs. Consequently, we take into account the following four types of LLMs:

1. ***Pre-trained Language Models:*** We use the term "pre-trained language models" (PLMs) to refer to those relatively small large language models, such as Bert [9] and Deberta [21], which can be fine-tuned for downstream tasks. It should be noted that strictly speaking, all LLMs can be viewed as PLMs. Here we adopt the commonly used terminology for models like BERT [51] to distinguish them from other LLMs follwing the convention in a recent paper [85].
2. ***Deep Sentence Embedding Models:*** These models typically use PLMs as the base encoders and adopt the bi-encoder structure [54; 68; 44]. They further pre-train the models in a supervised [54] or contrastive manner [68; 44]. In most cases, there is no need for these models to conduct additional fine-tuning for downstream tasks. These models can be further categorized into *local sentence embedding models* and *online sentence embedding models*. *Local sentence embedding models* are open-source and can be accessed locally, with Sentence-BERT (SBERT) being an example. On the other hand, *online sentence embedding models* are closed-source and deployed as services, with OpenAI's text-ada-embedding-002 [44] being an example.
3. ***Large Language Models:*** Compared to PLMs, Large Language Models (LLMs) exhibit significantly enhanced capabilities with orders of magnitude more parameters. LLMs can be categorized into two types. The first type consists of open-source LLMs, which can be deployed locally, providing users with transparent access to the models' parameters and embeddings. However, the substantial size of these models poses a challenge, as fine-tuning them can be quite cumbersome. One representative example of an open-source LLM is LLaMA [63]. The second type of LLMs is typically deployed as services [59], with restrictions placed on user interfaces. In this case, users are unable to access the model parameters, embeddings, or logits directly. The most powerful LLMs such as ChatGPT [45] and GPT4 [46] belong to this kind.

Among the four types of LLMs, PLMs, deep sentence embedding models, and open-source LLMs are often embedding-visible LLMs. Closed-source LLMs are embedding-invisible LLMs.

3. PIPELINES FOR LLMS IN GRAPHS

Given the superior power of LLMs in understanding textual information, we now investigate different strategies to leverage LLMs for node classification in textual graphs. Specifically, we present two distinct pipelines: *LLMs-as-Enhancers* and *LLMs-as-Predictors*. Figure 1 provides figurative illustrations of these two pipelines, and we elaborate on their details as follows.

LLMs-as-Enhancers In this pipeline, LLMs are leveraged to enhance the text attributes. As shown in Figure 1, for *LLMs-as-Enhancers*, LLMs are adopted to pre-process the text attributes, and then GNNs are trained on the enhanced attributes as the predictors. Considering different structures of LLMs, we conduct enhancements either at the **feature level** or at the **text level** as shown in Figure 2.

1. ***Feature-level enhancement:*** For feature-level enhancement, embedding-visible LLMs inject their knowledge by simply encoding the text attribute s_i into text embeddings $h_i \in R^d$. We investigate two feasible **integrating structures** for feature-level enhancement. **(1) Cascading structure:** Embedding-visible LLMs and GNNs are combined sequentially. Embedding-visible LLMs first encode text attributes into text features, which are then adopted as the initial node features for GNNs. **(2) Iterative structure [83]:** PLMs and GNNs are co-trained together by generating pseudo labels for each other. Only PLMs are suitable for this structure since it involves fine-tuning.
2. ***Text-level enhancement:*** For text-level enhancement, given the text attribute s_i, LLMs will first transform the text attribute into augmented attribute s_i^{Aug}. Enhanced attributes will then be encoded into enhanced node features $h_i^{Aug} \in R^d$ through embedding-visible LLMs. GNNs will make predictions by ensembling the original node features and augmented node features.

LLMs-as-Predictors In this pipeline, LLMs are leveraged to directly make predictions for the node classification task. As shown in Figure 1b, for *LLMs-as-Predictors*, the first step is to design prompts to represent graph structural information, text attributes, and label information with texts. Then, embedding-invisible LLMs make predictions based on the information embedded in the prompts.

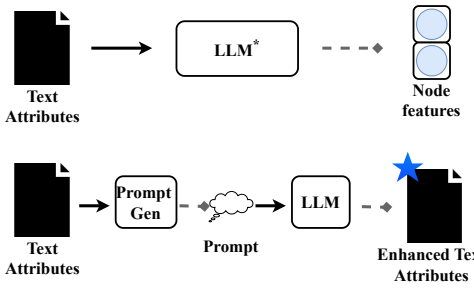
4. LLMS AS THE ENHANCERS

In this section, we investigate the potential of employing LLMs to enrich the text attributes of nodes. As presented in Section 3, we consider ***feature-level enhancement***, which injects LLMs' knowledge by encoding text attributes into features. Moreover, we consider ***text-level enhancement***, which inject LLMs' knowledge by augmenting the text attributes at the text level. We first study ***feature-level enhancement***.

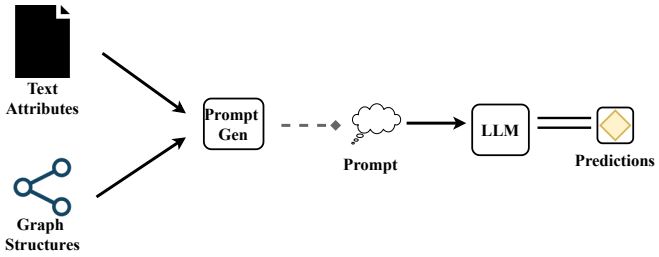
4.1 Feature-level Enhancement

In *feature-level enhancement*, we mainly study how to combine embedding-visible LLMs with GNNs at the feature level. The embedding generated by LLMs will be adopted as the initial features of GNNs. We first briefly introduce the dataset and dataset split settings we use.

Datasets. In this study, we adopt CORA [40], PUBMED [57], OGBN-ARXIV, and OGBN-PRODUCTS [23], four popular benchmarks for node classification. We present their detailed statistics and descriptions in Appendix A. Specifically, we



(a) *An illustration of LLMs-as-Enhancers*, where LLMs preprocess the text attributes, and GNNs eventually make the predictions. Three different structures for this pipeline are demonstrated in Figure 2.



(b) *An illustration of LLMs-as-Predictors*, where LLMs directly make the predictions. The key component for this pipeline is how to design an effective prompt to incorporate structural and attribute information.

Figure 1: Pipelines for integrating LLMs into graph learning. In all figures, we use "PLM" to denote small-scale PLMs that can be fine-tuned on downstream datasets, "LLM*" to denote embedding-visible LLMs, and "LLM" to denote embedding-invisible LLMs.

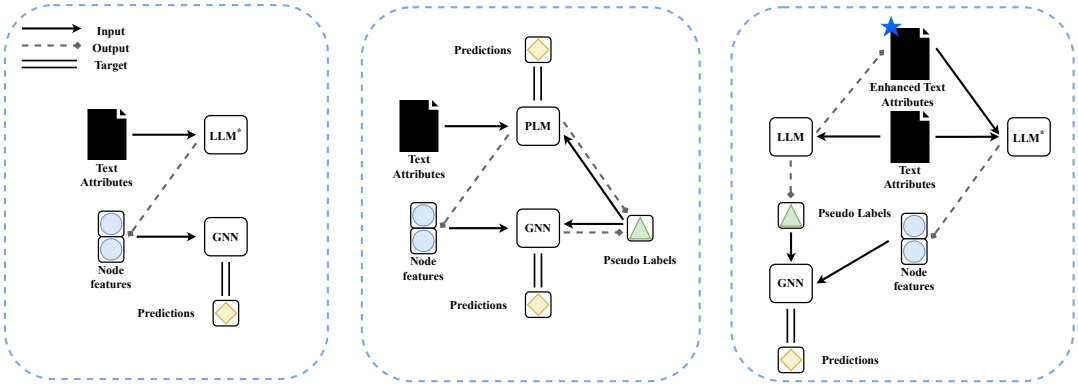


Figure 2: Three strategies to adopt LLMs as enhancers. The first two integrating structures are designed for feature-level enhancement, while the last structure is designed for text-level enhancement. From left to right: (1) Cascading Structure: Embedding-visible LLMs enhance text attributes directly by encoding them into initial node features for GNNs. (2) Iterative Structure: GNNs and PLMs are co-trained in an iterative manner. (3) Text-level enhancement structure: Embedding-invisible LLMs are initially adopted to enhance the text attributes by generating augmented attributes. The augmented attributes and original attributes are encoded and then ensembled together.

examine two classification dataset split settings, specifically tailored for the CORA and PUBMED datasets. Meanwhile, for OGBN-ARXIV and OGBN-PRODUCTS, we adopt the official dataset splits. (1) For CORA and PUBMED, the first splitting setting addresses **low-labeling-rate** conditions, which is a commonly adopted setting [75]. To elaborate, we randomly select 20 nodes from each class to form the training set. Then, 500 nodes are chosen for the validation set, while 1000 additional random nodes from the remaining pool are used for the test set. (2) The second splitting setting caters to **high-labeling-rate** scenarios, which is also a commonly used setting, and also adopted by TAPE [22]. In this setting, 60% of the nodes are designated for the training set, 20% for the validation set, and the remaining 20% are set aside for the test set. We take the output of GNNs and compare it with the ground truth of the dataset. We conduct all the experiments on 10 different seeds and report both average accuracy and variance.

Baseline Models. In our exploration of how LLMs augment node attributes at the feature level, we consider three main components: (1) *Selection of GNNs*, (2) *Selection of LLMs*, and (3) *Intergrating structures for LLMs and GNNs*. In this study, we choose the most representative models for each component, and the details are listed below.

1. *Selection of GNNs:* For GNNs on CORA and PUBMED, we consider Graph Convolutional Network (GCN) [27] and Graph Attention Network (GAT) [64]. We also include the performance of MLP to **evaluate the quality of text embeddings without aggregations**. For OGBN-ARXIV, we consider GCN, MLP, and a better-performed GNN model RevGAT [28]. For OGBN-PRODUCTS, we consider GraphSAGE [19] which supports neighborhood sampling for large graphs, MLP, and a state-of-the-art model SAGN [58]. For RevGAT and SAGN, we adopt all tricks utilized in the OGB leaderboard [23][1].
2. *Selection of LLMs:* To enhance the text attributes at the feature level, we specifically require embedding-visible LLMs. Specifically, we select (1) **Fixed PLM/LLMs without fine-tuning:** We consider Deberta [21] and LLaMA [63]. The first one is adapted from GLEM [83] and we follow the setting of GLEM [83] to adopt the

[1] https://ogb.stanford.edu/docs/leader_nodeprop/

[CLS] token of PLMs as the text embeddings. LLaMA is a widely adopted open-source LLM, which has also been included in Langchain[2]. We adopt LLaMA-cpp[3], which adopt the [EOS] token as text embeddings in our experiments. (2) **Local sentence embedding models**: We adopt Sentence-BERT [54] and e5-large [68]. The former is one of the most popular lightweight deep text embedding models while the latter is the state-of-the-art model on the MTEB leaderboard [43]. (3) **Online sentence embedding models**: We consider two online sentence embedding models, i.e., text-ada-embedding-002 [44] from OpenAI, and Palm-Cortex-001 [1] from Google. Although the strategy to train these models has been discussed [1; 44], their detailed parameters are not known to the public, together with their capability on node classification tasks. (4) **Fine-tuned PLMs**: We consider fine-tuning Deberta on the downstream dataset, and also adopt the last hidden states of PLMs as the text embeddings. For fine-tuning, we consider two integrating structures below.

3. *Integration structures:* We consider **cascading structure** and **iterative structure**. (1) **Cascading structure:** we first fine-tune the PLMs on the downstream dataset. Subsequently, the text embeddings engendered by the fine-tuned PLM are employed as the initial node features for GNNs. (2) **Iterative structure:** PLMs and GNNs are first trained separately and further co-trained in an iterative manner by generating pseudo labels for each other. This grants us the flexibility to choose either the final iteration of PLMs or GNNs as the predictive models, which are denoted as "GLEM-LM" and "GLEM-GNN", respectively.

We also consider non-contextualized shallow embeddings [41] including TF-IDF and Word2vec [23] as a comparison. TF-IDF is adopted to process the original text attributes for Pubmed [57], and Word2vec is utilized to encode the original text attributes for Ogbn-arxiv [23]. For Ogbn-arxiv and Ogbn-products, we also consider the GIANT features [6], which can not be directly applied to Cora and Pubmed because of its special pre-training strategy. Furthermore, we don't include LLaMA for Ogbn-arxiv and Ogbn-products because it imposes an excessive computational burden when dealing with large-scale datasets.

The results are shown in Table 1, Table 2, and Table 3. In these tables, we demonstrate the performance of different combinations of text encoders and GNNs. We also include the performance of MLPs which can suggest the original quality of the textual embeddings before the aggregation. Moreover, We use colors to show the top 3 best LLMs under each GNN (or MLP) model. Specifically, We use yellow to denote the best one under a specific GNN/MLP model, green the second best one, and pink the third best one.

4.1.1 Node Classification Performance Comparison

Observation 1. Combined with different types of text embeddings, GNNs demonstrate distinct effectiveness.

From Table 3, if we compare the performance of TF-IDF and fine-tuned PLM embeddings when MLP is the predictor, we can see that the latter usually achieves much better performance. However, when a GNN model is adopted as the predictor, the performance of TF-IDF embedding is close to and even surpasses the PLM embedding. This result is consistent with the findings in [49], which suggests that GNNs present distinct effectiveness for different types of text embeddings. However, we don't find a simple metric to determine the effectiveness of GNNs on different text embeddings. We will further discuss this limitation in Section 7.2.

Observation 2. Fine-tune-based LLMs may fail at low labeling rate settings.

From Table 1, we note that no matter the cascading structure or the iterative structure, fine-tune-based LLMs' embeddings perform poorly for low labeling rate settings. Both fine-tuned PLM and GLEM present a large gap against deep sentence embedding models and TF-IDF, which do not involve fine-tuning. When training samples are limited, fine-tuning may fail to transfer sufficient knowledge for the downstream tasks.

Observation 3. With a simple cascading structure, the combination of deep sentence embedding with GNNs makes a strong baseline.

From Table 1, Table 2, Table 3, we can see that with a simple cascading structure, the combination of deep sentence embedding models (including both local sentence embedding models and online sentence embedding models) with GNNs show competitive performance, under all dataset split settings. The intriguing aspect is that, during the pre-training stage of these deep sentence embedding models, no structural information is incorporated. Therefore, it is astonishing that these structure-unaware models can outperform GIANT on Ogbn-arxiv, which entails a structure-aware self-supervised learning stage.

Observation 4. Simply enlarging the model size of LLMs may not help with the node classification performance.

From Table 1 and Table 2, we can see that although the performance of the embeddings generated by LLaMA outperforms the Deberta-base without fine-tuning by a large margin, there is still a large performance gap between the performance of embeddings generated by deep sentence embedding models in the low labeling rate setting. This result indicates that simply increasing the model size may not be sufficient to generate high-quality embeddings for node classification. The pre-training objective may be an important factor.

4.1.2 Scalability Investigation

In the aforementioned experimental process, we empirically find that in larger datasets like Ogbn-arxiv, methods like GLEM that require fine-tuning of the PLMs will take several orders of magnitude more time in the training stage than these that do not require fine-tuning. It presents a hurdle for these approaches to be applied to even larger datasets or scenarios with limited computing resources. To gain a more comprehensive understanding of the efficiency and scalability of different LLMs and integrating structures, we conduct an experiment to measure the running time and memory usage of different approaches. It should be noted that we mainly consider the scalability problem in the training stage, which is different from the efficiency problem in the inference stage.

In this study, we choose representative models from each type of LLMs, and each kind of integrating structure. For

[2] https://python.langchain.com/

[3] https://github.com/ggerganov/llama.cpp

Table 1: Experimental results for feature-level *LLMs-as-Enhancer* on CORA and PUBMED with a low labeling ratio. Since MLPs do not provide structural information, it is meaningless to co-train it with PLM (with their performance shown as N/A). We use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

	CORA			PUBMED		
	GCN	GAT	MLP	GCN	GAT	MLP
Non-contextualized Shallow Embeddings						
TF-IDF	81.99 ± 0.63	82.30 ± 0.65	67.18 ± 1.01	78.86 ± 2.00	77.65 ± 0.91	71.07 ± 0.78
Word2Vec	74.01 ± 1.24	72.32 ± 0.17	55.34 ± 1.31	70.10 ± 1.80	69.30 ± 0.66	63.48 ± 0.54
PLM/LLM Embeddings without Fine-tuning						
Deberta-base	48.49 ± 1.86	51.02 ± 1.22	30.40 ± 0.57	62.08 ± 0.06	62.63 ± 0.27	53.50 ± 0.43
LLama 7B	66.80 ± 2.20	59.74 ± 1.53	52.88 ± 1.96	73.53 ± 0.06	67.52 ± 0.07	66.07 ± 0.56
Local Sentence Embedding Models						
Sentence-BERT(MiniLM)	82.20 ± 0.49	82.77 ± 0.59	74.26 ± 1.44	81.01 ± 1.32	79.08 ± 0.07	76.66 ± 0.50
e5-large	82.56 ± 0.73	81.62 ± 1.09	74.26 ± 0.93	82.63 ± 1.13	79.67 ± 0.80	80.38 ± 1.94
Online Sentence Embedding Models						
text-ada-embedding-002	82.72 ± 0.69	82.51 ± 0.86	73.15 ± 0.89	79.09 ± 1.51	80.27 ± 0.41	78.03 ± 1.02
Google Palm Cortex 001	81.15 ± 1.01	82.79 ± 0.41	69.51 ± 0.83	80.91 ± 0.19	80.72 ± 0.33	78.93 ± 0.90
Fine-tuned PLM Embeddings						
Fine-tuned Deberta-base	59.23 ± 1.16	57.38 ± 2.01	30.98 ± 0.68	62.12 ± 0.07	61.57 ± 0.07	53.65 ± 0.26
Iterative Structure						
GLEM-GNN	48.49 ± 1.86	51.02 ± 1.22	N/A	62.08 ± 0.06	62.63 ± 0.27	N/A
GLEM-LM	59.23 ± 1.16	57.38 ± 2.01	N/A	62.12 ± 0.07	61.57 ± 0.07	N/A

Table 2: Experimental results for feature-level *LLMs-as-Enhancers* on CORA and PUBMED with a high labeling ratio. We use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

	CORA			PUBMED		
	GCN	GAT	MLP	GCN	GAT	MLP
Non-contextualized Shallow Embeddings						
TF-IDF	90.90 ± 2.74	90.64 ± 3.08	83.98 ± 5.91	89.16 ± 1.25	89.00 ± 1.67	89.72 ± 3.57
Word2Vec	88.40 ± 2.25	87.62 ± 3.83	78.71 ± 6.32	85.50 ± 0.77	85.63 ± 0.93	83.80 ± 1.33
PLM/LLM Embeddings without Fine-tuning						
Deberta-base	65.86 ± 1.96	79.67 ± 3.19	45.64 ± 4.41	67.33 ± 0.69	67.81 ± 1.05	65.07 ± 0.57
LLama 7B	89.69 ± 1.86	87.66 ± 4.84	80.66 ± 7.72	88.26 ± 0.78	88.31 ± 2.01	89.39 ± 1.09
Local Sentence Embedding Models						
Sentence-BERT(MiniLM)	89.61 ± 3.23	90.68 ± 2.22	86.45 ± 5.56	90.32 ± 0.91	90.80 ± 2.02	90.59 ± 1.23
e5-large	90.53 ± 2.33	89.10 ± 3.22	86.19 ± 4.38	89.65 ± 0.85	89.55 ± 1.16	91.39 ± 0.47
Online Sentence Embedding Models						
text-ada-embedding-002	89.13 ± 2.00	90.42 ± 2.50	85.97 ± 5.58	89.81 ± 0.85	91.48 ± 1.94	92.63 ± 1.14
Google Palm Cortex 001	90.02 ± 1.86	90.31 ± 2.82	81.03 ± 2.60	89.78 ± 0.95	90.52 ± 1.35	91.87 ± 0.84
Fine-tuned PLM Embeddings						
Fine-tuned Deberta-base	85.86 ± 2.28	86.52 ± 1.87	78.20 ± 2.25	91.49 ± 1.92	89.88 ± 4.63	94.65 ± 0.13
Iterative Structure						
GLEM-GNN	89.13 ± 0.73	88.95 ± 0.64	N/A	92.57 ± 0.25	92.78 ± 0.21	N/A
GLEM-LM	82.71 ± 1.08	83.54 ± 0.99	N/A	94.36 ± 0.21	94.62 ± 0.14	N/A

TF-IDF, it's a shallow embedding that doesn't involve either training or inference, so the time and memory complexity of the LM phase can be neglected. In terms of Sentence-BERT, for the LM phase, this kind of local sentence embedding model does not involve a fine-tuning stage, and they only need to generate the initial embeddings. For text-ada-embedding-002, which is offered as an API service, we make API calls to generate embeddings. In this part, we set the batch size of Ada to 1,024 and call the API asynchronously, then we measure the time consumption to generate embeddings as the LM phase running time. For Deberta-base, we record the time used to fine-tune the model and generate the text embeddings as the LM phase running time. For GLEM, since it co-trains the PLM and GNNs, we consider LM phase running time and GNN phase running time together (and show the total training time in the "LM phase" column). The efficiency results are shown in Table 4. We also report the peak memory usage in the table. We adopt the default output dimension of each text encoder, which is shown in the brackets.

Observation 5. For integrating structures, iterative structure introduces massive computation overhead in the training stage.

From Table 2 and Table 3, GLEM presents a superior performance in datasets with an adequate number of labeled training samples, especially in large-scale datasets like OGBN-ARXIV and OGBN-PRODUCTS. However, from Table 4, we can see that it introduces massive computation overhead in the training stage compared to Deberta-base with a cascading structure, which indicates the potential efficiency problem of the iterative structures.

Moreover, from Table 4, we note that for the GNN phase,

Table 3: Experimental results for feature-level *LLMs-as-Enhancers* on OGBN-ARXIV and OGBN-PRODUCTS dataset. MLPs do not provide structural information so it's meaningless to co-train it with PLM, thus we don't show the performance. We use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

	OGBN-ARXIV			OGBN-PRODUCTS		
	GCN	MLP	RevGAT	SAGE	SAGN	MLP
Non-contextualized Shallow Embeddings						
TF-IDF	72.23 ± 0.21	66.60 ± 0.25	75.16 ± 0.14	79.73 ± 0.48	84.40 ± 0.07	64.42 ± 0.18
Word2Vec	71.74 ± 0.29	55.50 ± 0.23	73.78 ± 0.19	81.33 ± 0.79	84.12 ± 0.18	69.27 ± 0.54
PLM/LLM Embeddings without Fine-tuning						
Deberta-base	45.70 ± 5.59	40.33 ± 4.53	71.20 ± 0.48	62.03 ± 8.82	74.90 ± 0.48	7.18 ± 1.09
Local Sentence Embedding Models						
Sentence-BERT(MiniLM)	73.10 ± 0.25	71.62 ± 0.10	76.94 ± 0.11	82.51 ± 0.53	84.79 ± 0.23	72.73 ± 0.34
e5-large	73.74 ± 0.12	72.75 ± 0.00	76.59 ± 0.44	82.46 ± 0.91	85.47 ± 0.21	77.49 ± 0.29
Online Sentence Embedding Models						
text-ada-embedding-002	72.76 ± 0.23	72.17 ± 0.00	76.64 ± 0.20	82.90 ± 0.42	85.20 ± 0.19	76.42 ± 0.31
Fine-tuned PLM Embeddings						
Fine-tuned Deberta-base	74.65 ± 0.12	72.90 ± 0.11	75.80 ± 0.39	82.15 ± 0.16	84.01 ± 0.05	79.08 ± 0.23
Others						
GIANT	73.29 ± 0.10	73.06 ± 0.11	75.90 ± 0.19	83.16 ± 0.19	86.67 ± 0.09	79.82 ± 0.07
Iterative Structure						
GLEM-GNN	75.93 ± 0.19	N/A	76.97 ± 0.19	83.16 ± 0.09	87.36 ± 0.07	N/A
GLEM-LM	75.71 ± 0.24	N/A	75.45 ± 0.12	81.25 ± 0.15	84.83 ± 0.04	N/A

Table 4: Efficiency analysis on OGBN-ARXIV. Note that we show the dimension of generated embeddings in the brackets. For GIANT, it adopts a special pre-training stage, which will introduce computation overhead with orders of magnitude larger than that of fine-tuning. The specific time was not discussed in the original paper, therefore its cost in LM-phase is not shown in the table.

Input features	Backbone	LM-phase Running time(s)	LM-phase Memory (GB)	GNN-phase Running time(s)	GNN-phase Memory (GB)
TF-IDF (1024)	GCN	N/A	N/A	53	9.81
	RevGAT	N/A	N/A	873	7.32
Sentence-BERT (384)	GCN	239	1.30	48	7.11
	RevGAT	239	1.30	674	4.37
text-ada-embedding-002 (1536)	GCN	165	N/A	73	11.00
	RevGAT	165	N/A	1038	8.33
Deberta-base (768)	GCN	13560	12.53	50	9.60
	RevGAT	13560	12.53	122	6.82
GLEM-GNN (768)	GCN	68071	18.22	N/A	N/A
	RevGAT	68294	18.22	N/A	N/A
GIANT (768)	GCN	N/A	N/A	50	9.60
	RevGAT	N/A	N/A	122	6.82

the dimension of initial node features, which is the default output dimension of text encoders mainly determines memory usage and time cost.

Observation 6. In terms of different LLM types, deep sentence embedding models present better efficiency in the training stage.

In Table 4, we analyze the efficiency of different types of LLMs by selecting representative models from each category. Comparing fine-tune-based PLMs with deep sentence embedding models, we observe that the latter demonstrates significantly better time efficiency as they do not require a fine-tuning stage. Additionally, deep sentence embedding models exhibit improved memory efficiency as they solely involve the inference stage without the need to store additional information such as gradients.

4.2 Text-level Enhancement

For feature-level enhancement, LLMs in the pipeline must be embedding-visible. However, the most powerful LLMs such as ChatGPT [45], PaLM [1], and GPT4 [46] are all deployed as online services [59], which put strict restrictions so that users can not get access to model parameters and embeddings. Users can only interact with these embedding-invisible LLMs through texts, which means that user inputs must be formatted as texts and LLMs will only yield text outputs. In this section, we explore the potential for these embedding-invisible LLMs to do text-level enhancement. To enhance the text attribute at the text level, the key is to expand more information that is not contained in the original text attributes. Based on this motivation and a recent paper [22], we study the following two potential text-level enhancements, and illustrative examples of these two augmentations are shown in Figure 3.

1. **TAPE** [22]: The motivation of TAPE is to leverage the knowledge of LLMs to generate high-quality node features. Specifically, it uses LLMs to generate pseudo labels

and explanations. These explanations aim to make the logical relationship between the text features and corresponding labels more clear. For example, given the original attributes "mean-field approximation" and the ground truth label "probabilistic methods", it will generate a description such as "mean-field approximation is a widely adopted simplification technique for probabilistic models", which makes the connection of these two attributes much more clear. After generating pseudo labels and explanations, they further adopt PLMs to be fine-tuned on both the original text attributes and the explanations generated by LLMs, separately. Next, they generate the corresponding text features and augmented text features based on the original text attributes and augmented text attributes respectively, and finally ensemble them together as the initial node features for GNNs.

2. **Knowledge-Enhanced Augmentation**: The motivation behind Knowledge-Enhanced Augmentation (KEA) is to enrich the text attributes by providing additional information. KEA is inspired by knowledge-enhanced PLMs such as ERNIE [61] and K-BERT [36] and aims to explicitly incorporate external knowledge. In KEA, we prompt the LLMs to generate a list of knowledge entities along with their text descriptions. For example, we can generate a description for the abstract term "Hopf-Rinow theorem" as follows: "The Hopf-Rinow theorem establishes that a Riemannian manifold, which is both complete and connected, is geodesically complete if and only if it is simply connected." By providing such descriptions, we establish a clearer connection between the theorem and the category "Riemannian geometry". Once we obtain the entity list, we encode it either together with the original text attribute or separately. We try encoding text attributes with fine-tuned PLMs and deep sentence embedding models. We also employ ensemble methods to combine these embeddings. One potential advantage of KEA is that it is loosely coupled with the prediction performance of LLMs. In cases where LLMs generate incorrect predictions, TAPE may potentially generate low-quality node features because the explanations provided by PLMs may also be incorrect. However, with KEA, the augmented features may exhibit better stability since we do not rely on explicit predictions from LLMs.

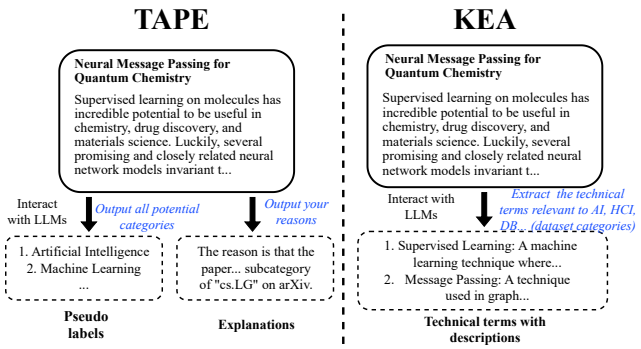


Figure 3: Illustrations for TAPE and KEA. TAPE leverages the knowledge of LLMs to generate explanations for their predictions. For KEA, we prompt the LLMs to generate a list of technical terms with their descriptions. The main motivation is to augment the attribute information.

4.2.1 *Experimental Setups*

To evaluate these two strategies, we conduct experiments on two small datasets CORA and PUBMED considering the cost to use the LLMs. For low labeling ratio and high labeling ratio, we adopt the same setting as that in Table 1 and Table 2. For predictors, we adopt GCN, GAT, and MLP to study both the quality of textual embeddings before and after aggregations. For LLMs, we adopt ChatGPT with the latest version (gpt-3.5-turbo-0613). To better understand the effectiveness of TAPE, we separate it into TA, P, and E, where "TA" refers to "text attributes", "P" refers to "pseudo labels", and "E" refers to "explanations". For KEA, we try two approaches to inject the augmented textual attributes. The first approach is appending the augmented textual attributes into the original attribute, which is denoted as "KEA-I". Then the combined attributes are encoded into features. The second approach is to encode the augmented attributes and original attributes separately, which is denoted as "KEA-S". We report the results for original, augmented, and ensembling features. Both TAPE and KEA adopt the cascading structures. After encoding the text attributes with LLMs, the generated embeddings are adopted as the initial features for GNNs. We try two approaches to encode the attributes, which are fine-tuned PLMs and local sentence embedding models. Specifically, we adopt Deberta-base and e5-large. To conduct a fair comparison, we first determine the better text encoder by evaluating their overall performance. Once the text encoder is selected, we proceed to compare the performance of the augmented attributes against the original attributes.

A comprehensive evaluation of TAPE. We first gain a deeper understanding of TAPE through a comprehensive ablation study. The experimental results are shown in Table 5 and Table 6. We show the approach we adopt to encode the text attributes in the bracket. In particular, we mainly consider fine-tuned Deberta-base, which is denoted as PLM, and e5-large, which is denoted as e5.

Observation 7. The effectiveness of TAPE is mainly from the explanations E generated by LLMs.

From the ablation study, we can see that compared to pseudo labels **P**, the explanations present better stability across different datasets. One main advantage of adopting explanations generated by LLMs is that these augmented attributes present better performance in the low-labeling rate setting. From Table 5, we note that when choosing PLM as the encoders, **E** performs much better than **TA** in the low labeling rate setting. Compared to explanations, we find that the effectiveness of the P mainly depends on the zero-shot performance of LLMs, which may present large variances across different datasets. In the following analysis, we use **TA** + **E** and neglect the pseudo labels generated by LLMs.

Observation 8. Replacing fine-tuned PLMs with deep sentence embedding models can further improve the overall performance of TAPE.

From Table 5 and Table 6, we observe that adopting e5-large as the LLMs to encode the text attributes can achieve good performance across different datasets and different data splits. Specifically, the **TA** + **E** encoded with e5 can achieve top 3 performance in almost all settings. In the following analysis, we adopt e5 to encode the original and enhanced attributes **TA** + **E**.

Table 5: A detailed ablation study of TAPE on CORA and PUBMED dataset in low labeling rate setting. For each combination of features and models, we use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

		CORA			PUBMED		
		GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
TAPE	**TAPE**	74.56 ± 2.03	75.27 ± 2.10	64.44 ± 0.60	85.97 ± 0.31	86.97 ± 0.33	93.18 ± 0.28
	P	52.79 ± 1.47	62.13 ± 1.50	63.56 ± 0.52	81.92 ± 1.89	88.27 ± 0.01	93.27 ± 0.15
	TA + E (e5)	83.38 ± 0.42	84.00 ± 0.09	75.73 ± 0.53	87.44 ± 0.49	86.71 ± 0.92	90.25 ± 1.56
	TA + E (PLM)	78.02 ± 0.56	64.08 ± 12.36	55.72 ± 11.98	80.70 ± 1.73	79.66 ± 3.08	76.42 ± 2.18
	E (PLM)	79.46 ± 1.10	74.82 ± 1.19	63.04 ± 0.88	81.88 ± 0.05	81.56 ± 0.07	76.90 ± 1.60
	E (e5)	84.38 ± 0.36	83.01 ± 0.60	70.64 ± 1.10	82.23 ± 0.78	80.30 ± 0.77	77.23 ± 0.48
Original attributes	**TA** (PLM)	59.23 ± 1.16	57.38 ± 2.01	30.98 ± 0.68	62.12 ± 0.07	61.57 ± 0.07	53.65 ± 0.26
	TA (e5)	82.56 ± 0.73	81.62 ± 1.09	74.26 ± 0.93	82.63 ± 1.13	79.67 ± 0.80	80.38 ± 1.94

Table 6: A detailed ablation study of TAPE on CORA and PUBMED dataset in the high labeling rate setting. For each combination of features and models, we use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

		CORA			PUBMED		
		GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
TAPE	**TAPE**	87.88 ± 0.98	88.69 ± 1.13	83.09 ± 0.91	92.22 ± 1.30	93.35 ± 1.50	95.05 ± 0.27
	P	64.90 ± 1.39	80.11 ± 4.01	70.31 ± 1.91	85.73 ± 0.59	91.60 ± 0.62	93.65 ± 0.35
	TA + E (e5)	90.68 ± 2.12	91.86 ± 1.36	87.00 ± 4.83	92.64 ± 1.00	93.35 ± 1.24	94.34 ± 0.86
	TA + E (PLM)	87.44 ± 1.74	88.40 ± 1.60	82.80 ± 1.00	90.23 ± 0.71	91.73 ± 1.58	95.40 ± 0.32
	E (PLM)	83.28 ± 4.53	82.47 ± 6.06	80.41 ± 3.35	88.90 ± 2.94	83.00 ± 14.07	87.75 ± 14.83
	E (e5)	89.39 ± 2.69	90.13 ± 2.52	84.05 ± 4.03	89.68 ± 0.78	90.61 ± 1.61	91.09 ± 0.85
Original attributes	**TA** (PLM)	85.86 ± 2.28	86.52 ± 1.87	78.20 ± 2.25	91.49 ± 1.92	89.88 ± 4.63	94.65 ± 0.13
	TA (e5)	90.53 ± 2.33	89.10 ± 3.22	86.19 ± 4.38	89.65 ± 0.85	89.55 ± 1.16	91.39 ± 0.47

4.2.1.1 *Effectiveness of* KEA.

We then show the results of **KEA** in Table 7 and Table 8. For **KEA-I**, we inject the description of each technical term directly into the original attribute. For **KEA-S**, we encode the generated description and original attribute separately.

Observation 9. The proposed knowledge enhancement attributes KEA can enhance the performance of the original attribute TA.

From Table 7 and Table 8, we first compare the performance of features encoded by e5 and PLM. We see that the proposed **KEA** is more fitted to the e5 encoder, and fine-tuned PLM embeddings present poor performance on the low labeling rate, thus we also select e5 as the encoder to further compare the quality of attributes. From Table 9 we can see that the proposed **KEA-I + TA** and **KEA-S + TA** attributes can consistently outperform the original attributes **TA**.

Observation 10. For different datasets, the most effective enhancement methods may vary.

Moreover, we compare the performance of our proposed **KEA** with **TA + E**, and the results are shown in Table 9. We can see that on CORA, our methods can achieve better performance while **TA + E** can achieve better performance on PUBMED. One potential explanation for this phenomenon is that **TA + E** relies more on the capability of LLMs. Although we have removed the pseudo labels **P**, we find that the explanations still contain LLMs' predictions. As a result, the effectiveness of **TA + E** will be influenced by LLMs' performance on the dataset. As shown in [22], the LLMs can achieve superior performance on the PUBMED dataset but perform poorly on the CORA dataset. Compared to **TA + E**, our proposed **KEA** only utilizes the commonsense knowledge of the LLMs, which may have better stability across different datasets.

5. LLMS AS THE PREDICTORS

In the *LLMs-as-Enhancers* pipeline, the role of the LLMs remains somewhat limited since we only utilize their pre-trained knowledge but overlook their reasoning capability. Drawing inspiration from the LLMs' proficiency in handling complex tasks with implicit structures, such as logical reasoning [7] and recommendation [14], we question: **Is it possible for the LLM to independently perform predictive tasks on graph structures?** By shifting our focus to node attributes and overlooking the graph structures, we can perceive node classification as a text classification problem. In [60], the LLMs demonstrate significant promise, suggesting that they can proficiently process text attributes. However, one key problem is that LLMs are not originally designed to process graph structures. Therefore, it can not directly process structural information like GNNs.

In this section, we aim to explore the potential of LLMs as a predictor. In particular, we first check whether LLM can perform well without any structural information. Then, we further explore some prompts to incorporate structural information with natural languages. Finally, we show a case study in Section 5.3 to explore its potential usage as an annotator for graphs.

5.1 How Can LLM Perform on Popular Graph Benchmarks without Structural Information?

In this subsection, we treat the node classification problem as a text classification problem by ignoring the structural information. We adopt ChatGPT (gpt-3.5-turbo-0613) as the LLMs to conduct all the experiments. We choose five popular textual graph datasets with raw text attributes: CORA [40], CITESEER [15], PUBMED [57], OGBN-ARXIV, and OGBN-PRODUCTS [23]. The details of these datasets can be found in Appendix A. Considering the costs to query LLMs'

Table 7: A detailed ablation study of KEA on Cora and Pubmed dataset in the low labeling rate setting. For each combination of features and models, we use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

		Cora			Pubmed		
		GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
Original attributes	TA (PLM)	59.23 ± 1.16	57.38 ± 2.01	30.98 ± 0.68	62.12 ± 0.07	61.57 ± 0.07	53.65 ± 0.26
	TA (e5)	82.56 ± 0.73	81.62 ± 1.09	74.26 ± 0.93	82.63 ± 1.13	79.67 ± 0.80	80.38 ± 1.94
KEA	KEA-I + TA (e5)	83.20 ± 0.56	83.38 ± 0.63	74.34 ± 0.97	83.30 ± 1.75	81.16 ± 0.87	80.74 ± 2.44
	KEA-I + TA (PLM)	53.21 ± 11.54	55.38 ± 4.64	31.80 ± 3.63	57.13 ± 8.20	58.66 ± 4.27	52.28 ± 4.47
	KEA-I (e5)	81.35 ± 0.77	82.04 ± 0.72	70.64 ± 1.10	81.98 ± 0.91	81.04 ± 1.39	79.73 ± 1.63
	KEA-I (PLM)	36.68 ± 18.63	37.69 ± 12.79	30.46 ± 0.60	56.22 ± 7.17	59.33 ± 1.69	52.79 ± 0.51
	KEA-S + TA (e5)	84.63 ± 0.58	85.02 ± 0.40	76.11 ± 2.66	82.93 ± 2.38	81.34 ± 1.51	80.74 ± 2.44
	KEA-S + TA (PLM)	51.36 ± 16.13	52.85 ± 7.00	34.56 ± 5.09	59.47 ± 6.09	51.93 ± 3.27	51.11 ± 2.63
	KEA-S (e5)	84.38 ± 0.36	83.01 ± 0.60	70.64 ± 1.10	82.23 ± 0.78	80.30 ± 0.77	77.23 ± 0.48
	KEA-S (PLM)	28.97 ± 18.24	43.88 ± 10.31	30.36 ± 0.58	61.22 ± 0.94	54.93 ± 1.55	47.94 ± 0.89

Table 8: A detailed ablation study of KEA on Cora and Pubmed dataset in the high labeling rate setting. For each combination of features and models, we use yellow to denote the best performance under a specific GNN/MLP model, green the second best one, and pink the third best one.

		Cora			Pubmed		
		GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
Original Attributes	**TA** (PLM)	85.86 ± 2.28	86.52 ± 1.87	78.20 ± 2.25	91.49 ± 1.92	89.88 ± 4.63	94.65 ± 0.13
	TA (e5)	90.53 ± 2.33	89.10 ± 3.22	86.19 ± 4.38	89.65 ± 0.85	89.55 ± 1.16	91.39 ± 0.47
KEA	**KEA-I + TA** (e5)	91.12 ± 1.76	90.24 ± 2.93	87.88 ± 4.44	90.19 ± 0.83	90.60 ± 1.22	92.12 ± 0.74
	KEA-I + TA (PLM)	87.07 ± 1.04	87.66 ± 0.86	79.12 ± 2.77	92.32 ± 0.64	92.29 ± 1.43	94.85 ± 0.20
	KEA-I (e5)	91.09 ± 1.78	90.13 ± 2.76	86.78 ± 4.12	89.56 ± 0.82	90.25 ± 1.34	91.92 ± 0.80
	KEA-I (PLM)	86.08 ± 2.35	85.23 ± 3.15	77.97 ± 2.87	91.73 ± 0.58	91.93 ± 1.76	94.76 ± 0.33
	KEA-S + TA (e5)	91.09 ± 1.78	92.30 ± 1.69	88.95 ± 4.96	90.40 ± 0.92	90.82 ± 1.30	91.78 ± 0.56
	KEA-S + TA (PLM)	83.98 ± 5.13	87.33 ± 1.68	80.04 ± 1.32	86.11 ± 5.68	89.04 ± 5.82	94.35 ± 0.48
	KEA-S (e5)	89.39 ± 2.69	90.13 ± 2.52	84.05 ± 4.03	89.68 ± 0.78	90.61 ± 1.61	91.09 ± 0.85
	KEA-S (PLM)	83.35 ± 7.30	85.67 ± 2.00	76.76 ± 1.82	79.68 ± 19.57	69.90 ± 19.75	85.91 ± 6.47

Table 9: Comparison of the performance of TA, KEA-I, and KEA-S, and TA + E. The best performance is shown with an underline. Cora (low) means a low labeling rate setting, and Cora (high) denotes a high labeling rate setting.

	Cora (low)			Pubmed (low)		
	GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
TA	82.56 ± 0.73	81.62 ± 1.09	74.26 ± 0.93	82.63 ± 1.13	79.67 ± 0.80	80.38 ± 1.94
KEA-I + TA	83.20 ± 0.56	83.38 ± 0.63	74.34 ± 0.97	83.30 ± 1.75	81.16 ± 0.87	80.74 ± 2.44
KEA-S + TA	84.63 ± 0.58	85.02 ± 0.40	76.11 ± 2.66	82.93 ± 2.38	81.34 ± 1.51	80.74 ± 2.44
TA+E	83.38 ± 0.42	84.00 ± 0.09	75.73 ± 0.53	87.44 ± 0.49	86.71 ± 0.92	90.25 ± 1.56
	Cora (high)			**Pubmed (high)**		
	GCN	**GAT**	**MLP**	**GCN**	**GAT**	**MLP**
TA	90.53 ± 2.33	89.10 ± 3.22	86.19 ± 4.38	89.65 ± 0.85	89.55 ± 1.16	91.39 ± 0.47
KEA-I + TA	91.12 ± 1.76	90.24 ± 2.93	87.88 ± 4.44	90.19 ± 0.83	90.60 ± 1.22	92.12 ± 0.74
KEA-S + TA	91.09 ± 1.78	92.30 ± 1.69	88.95 ± 4.96	90.40 ± 0.92	90.82 ± 1.30	91.78 ± 0.56
TA+E	90.68 ± 2.12	91.86 ± 1.36	87.00 ± 4.83	92.64 ± 1.00	93.35 ± 1.24	94.34 ± 0.86

APIs, it's not possible for us to test the whole dataset for these graphs. Considering the rate limit imposed by OpenAI[4], we randomly select 200 nodes from the test sets as our test data. In order to ensure that these 200 nodes better represent the performance of the entire set, we repeat all experiments twice. Additionally, we employ zero-shot performance as a sanity check, comparing it with the results in TAPE [22] to ensure minimal discrepancies.

We explore the following strategies:

1. **Zero-shot prompts**: This approach solely involves the attribute of a given node.
2. **Few-shot prompts**: On the basis of zero-shot prompts, few-shot prompts provide in-context learning samples together with their labels for LLMs to better understand the task. In addition to the node's content, this approach integrates the content and labels of randomly selected in-context samples from the training set. In the section, we adopt random sampling to select few-shot prompts.
3. **Zero-shot prompts with Chain-of-Thoughts (CoT)**: CoT [70] presents its effectiveness in various reasoning tasks, which can greatly improve LLMs' reasoning abilities. In this study, we test whether CoT can improve LLMs' capability on node classification tasks. On the basis of zero-shot prompts, we guide the LLMs to generate the thought process by using the prompt "think it step by step".
4. **Few-shot prompts with CoT**: Inspired by [82], which demonstrates that incorporating the CoT process generated by LLMs can further improve LLMs' reasoning

[4] https://platform.openai.com/docs/guides/rate-limits/overview

capabilities. Building upon the few-shot prompts, this approach enables the LLMs to generate a step-by-step thought process for the in-context samples. Subsequently, the generated CoT processes are inserted into the prompt as auxiliary information.

Output Parsing. In addition, we need a parser to extract the output from LLMs. We devise a straightforward approach to retrieve the predictions from the outputs. Initially, we instruct the LLMs to generate the results in a formatted output like "a python list". Then, we can use the symbols "[" and "]" to locate the expected outputs. It should be noted that this design aims to extract the information more easily but has little influence on the performance. We observe that sometimes LLMs will output contents that are slightly different from the expected format, for example, output the expected format "Information Retrieval" to "Information Extraction". In such cases, we compute the edit distance between the extracted output and the category names and select the one with the smallest distance. This method proves effective when the input context is relatively short. If this strategy encounters errors, we resort to extracting the first mentioned categories in the output texts as the predictions. If there's no match, then the model's prediction for the node is incorrect.

To reduce the variance of LLMs' predictions, we set the temperature to 0. For few-shot cases, we find that providing too much context will cause LLMs to generate outputs that are not compatible with the expected formats. Therefore, we set a maximum number of samples to ensure that LLMs generate outputs with valid formats. In this study, we choose this number to 2 and adopt accuracy as the performance metric.

5.1.1 Observations

Observation 11. LLMs present preliminary effectiveness on some datasets.

According to the results in Table 10, it is evident that LLMs demonstrate remarkable zero-shot performance on PUBMED. When it comes to OGBN-PRODUCTS, LLMs can achieve performance levels comparable to fine-tuned PLMs. However, there is a noticeable performance gap between LLMs and GNNs on CORA and PUBMED datasets. To gain a deeper understanding of this observation, it is essential to analyze the output of LLMs.

Observation 12. Wrong predictions made by LLMs are sometimes also reasonable.

After investigating the output of LLMs, we find that a part of the wrong predictions made by LLMs are very reasonable. An example is shown in Table 11. In this example, we can see that besides the ground truth label "Reinforcement Learning", "Neural Networks" is also a reasonable label, which also appears in the texts. We find that this is a common problem for CORA, CITESEER, and OGBN-ARXIV. For OGBN-ARXIV, there are usually multiple labels for one paper on the website. However, in the OGBN-ARXIV dataset, only one of them is chosen as the ground truth. This leads to a misalignment between LLMs' commonsense knowledge and the annotation bias inherent in these datasets. Moreover, we find that introducing few-shot samples presents little help to mitigate the annotation bias.

Observation 13. Chain-of-thoughts do not bring in performance gain.

For reasoning tasks in the general domain, chain-of-thoughts is believed to be an effective approach to increase LLM's reasoning capability [70]. However, we find that it's not effective for the node classification task. This phenomenon can be potentially explained by **Observation 12**. In contrast to mathematical reasoning, where a single answer is typically expected, multiple reasonable chains of thought can exist for node classification. An example is shown in Table 12. This phenomenon poses a challenge for LLMs as they may struggle to match the ground truth labels due to the presence of multiple reasonable labels.

Observation 14. For prompts that are very similar in semantics, there may be huge differences in their effects.

In addition, we observe that TAPE [22] implements a unique prompt on the OGBN-ARXIV dataset, yielding impressive results via zero-shot prompts. The primary distinction between their prompts and ours lies in the label design. Given that all papers originate from the computer science subcategory of Arxiv, they employ the brief term "arxiv cs subcategories" as a substitute for these 40 categories. Remarkably, this minor alteration contributes to a substantial enhancement in performance. To delve deeper into this phenomenon, we experiment with three disparate label designs: (1) Strategy 1: the original Arxiv identifier, such as "arxiv cs.CV"; (2) Strategy 2: natural language descriptors, like "computer vision"; and (3) Strategy 3: the specialized prompt, utilizing "arxiv cs subcategory" to denote all categories. Unexpectedly, we discover that Strategy 3 significantly outperforms the other two (refer to Table 13).

Given that LLMs undergo pre-training on extensive text corpora, it's likely that these corpora include papers from the Arxiv database. That specific prompt could potentially enhance the "activation" of these models' corresponding memory. However, the reason for the excellent results achieved by this kind of prompt might not stem from the simple data memorization of the LLM [25]. When applying to papers after 2023 that are not included in the pre-training corpus of the LLMs, this prompt also achieves similar effectiveness. This phenomenon reminds us that when using ChatGPT, sometimes providing more information in the prompt (such as category information from the OGBN-ARXIV dataset) may actually lead to a decrease in performance.

5.2 Incorporating Structural Information in the Prompts

As we note, LLMs can already present superior zero-shot performance on some datasets without providing any structural information. However, there is still a large performance gap between LLMs and GNNs in CORA, CITESEER, and OGBN-ARXIV. Then a question naturally raises that *whether we can further increase LLMs' performance by incorporating structural information?* To answer this problem, we first need to identify how to denote the structural information in the prompt. LLMs such as ChatGPT are not originally designed for graph structures, so they can not process adjacency matrices like GNNs. In this part, we study several ways to convey structural information and test their effectiveness on the CORA dataset.

Specifically, we first consider inputting the whole graph into the LLMs. Using CORA dataset as an example, we try to use prompts like "node 1: ⟨**paper content**⟩" to represent attributes, and prompts like "node 1 cites node 2" to represent the edge. However, we find that this approach is not

Table 10: Performance of LLMs on real-world text attributed graphs without structural information, we also include the result of GCN (or SAGE for OGBN-PRODUCTS) together with Sentence-BERT features. For CORA, CITESEER, PUBMED, we show the results of the low labeling rate setting.

	CORA	CITESEER	PUBMED	OGBN-ARXIV	OGBN-PRODUCTS
Zero-shot	67.00 ± 1.41	65.50 ± 3.53	90.75 ± 5.30	51.75 ± 3.89	70.75 ± 2.48
Few-shot	67.75 ± 3.53	66.00 ± 5.66	85.50 ± 2.80	50.25 ± 1.06	77.75 ± 1.06
Zero-shot with COT	64.00 ± 0.71	66.50 ± 2.82	86.25 ± 3.29	50.50 ± 1.41	71.25 ± 1.06
Few-shot with COT	64.00 ± 1.41	60.50 ± 4.94	85.50 ± 4.94	47.25 ± 2.47	73.25 ± 1.77
GCN/SAGE	82.20 ± 0.49	71.19 ± 1.10	81.01 ± 1.32	73.10 ± 0.25	82.51 ± 0.53

Table 11: A wrong but reasonable prediction made by LLMs

Paper: The Neural Network House: An overview; Typical home comfort systems utilize only rudimentary forms of energy management and conservation. The most sophisticated technology in common use today is an automatic setback thermostat. Tremendous potential remains for improving the efficiency of electric and gas usage...
Ground Truth: Reinforcement Learning
LLM's Prediction: Neural Networks

Table 12: An example that LLMs generate CoT processes not matching with ground truth labels

Paper: The Neural Network House: An overview.: Typical home comfort systems utilize only rudimentary forms of energy management and conservation. The most sophisticated technology in common use today is an automatic setback thermostat. Tremendous potential remains for improving the efficiency of electric and gas usage...
Generated Chain-of-thoughts: The paper discusses the use of neural networks for intelligent control and mentions the utilization of neural network reinforcement learning and prediction techniques. Therefore, the most likely category for this paper is 'Neural Networks'.
Ground Truth: Reinforcement Learning
LLM's Prediction: Neural Networks

Table 13: Performance of LLMs on OGB-Arxiv dataset, with three different label designs.

	Strategy 1	**Strategy 2**	**Strategy 3**
OGBN-ARXIV	48.5	51.8	74.5

feasible since LLMs usually present a small input context length restriction. As a result, we consider an "ego-graph" view, which refers to the subgraphs induced from the center nodes. In this way, we can narrow the number of nodes to be considered.

Specifically, we first organize the neighbors of the current nodes as a list of dictionaries consisting of attributes and labels of the neighboring nodes for training nodes. Then, the LLMs summarize the neighborhood information. It should be noted that we only consider 2-hop neighbors because GNNs typically have 2 layers, indicating that the 2-hop neighbor information is the most useful in most cases. Considering the input context limit of LLMs, we empirically find that each time we can summarize the attribute information of 5 neighbors. In this paper, we sample neighbors once and only summarize those selected neighbors. In practice, we can sample multiple times and summarize each of them to obtain more fine-grained neighborhood information.

Observation 15. Neighborhood summarization is likely to achieve performance gain.

From Table 14, we note that incorporating neighborhood information in either zero-shot or few-shot approaches yields performance gains compared to the zero-shot prompt without structural information except on the PUBMED dataset. By following the "homophily" assumption [87; 39], which suggests that neighboring nodes tend to share the same labels, the inclusion of neighboring information can potentially alleviate annotation bias. For instance, let's consider a paper from Arxiv covering general topics like transformers. Merely analyzing the content of this paper makes it difficult to determine which category the author would choose, as categories such as "Artificial Intelligence," "Machine Learning," and "Computer Vision" are all plausible options. However, by examining its citation relationships, we can better infer the author's bias. If the paper cites numerous sources from the "Computer Vision" domain, it is likely that the author is also a researcher in that field, thereby favoring the selection of this category. Consequently, structural information provides implicit supervision to assist LLMs in capturing the inherent annotation bias in the dataset. However, from the PUBMED dataset, we observe that incorporating neighborhood information results in clear performance drop, which necessitates a deep analysis below.

Observation 16. LLMs with structure prompts may suffer from heterophilous neighboring nodes.

From Table 14, we observe that LLMs perform worse on PUBMED after incorporating the structural information. To gain a deeper understanding, we focus on those nodes where zero-shot prompts without structural information can lead to correct prediction but prompts with 2-hop information can't.

An example of this kind of node is shown in Table 15. After analyzing the 2-hop neighbors of this node, we find that 15 out of 19 2-hop neighboring nodes have different labels against this node. This case is usually denoted as "heterophily" [87], which is a phenomenon in graph theory where nodes in a graph tend to connect with nodes that are dissimilar to them. In this case, we find that both GNNs and LLMs with a structure-aware prompt make wrong predictions. However, LLMs ignoring structural information get correct predictions, which indicates that LLMs with a structure-aware prompt may also suffer from the "het-

Table 14: Performance of LLMs on real-world text attributed graphs with summarized neighborhood information. For CORA, CITESEER, PUBMED, we show the results of the low labeling rate setting. We also include the result of GCN (or SAGE for OGBN-PRODUCTS) together with Sentence-BERT features.

	CORA	CITESEER	PUBMED	OGBN-ARXIV	OGBN-PRODUCTS
Zero-shot	67.00 ± 1.41	65.50 ± 3.53	90.75 ± 5.30	51.75 ± 3.89	70.75 ± 2.48
Few-shot	67.75 ± 3.53	66.00 ± 5.66	85.50 ± 2.80	50.25 ± 1.06	77.75 ± 1.06
Zero-Shot with 2-hop info	71.75 ± 0.35	62.00 ± 1.41	88.00 ± 1.41	55.00 ± 2.83	75.25 ± 3.53
Few-Shot with 2-hop info	74.00 ± 4.24	67.00 ± 4.94	79.25 ± 6.71	52.25 ± 3.18	76.00 ± 2.82
GCN/SAGE	82.20 ± 0.49	71.19 ± 1.10	81.01 ± 1.32	73.10 ± 0.25	82.51 ± 0.53

erophily" problem.

Table 15: GNNs and LLMs with structure-aware prompts are both wrong

Paper: Title: C-reactive protein and incident cardiovascular events among men with diabetes.
Abstract: OBJECTIVE: Several large prospective studies have shown that baseline levels of C-reactive protein (CRP) ...
Neighbor Summary: This paper focuses on different aspects of **type 2 diabetes** mellitus. It explores the levels of various markers such as tumor necrosis factor-alpha, interleukin-2 ...
Ground truth: "Diabetes Mellitus Type 1"
Structure-ignorant prompts: "Diabetes Mellitus Type 1"
Structure-aware prompt: "Diabetes Mellitus Type 2"
GNN: "Diabetes Mellitus Type 2"

5.3 Case Study: LLMs as the Pseudo Annotators

From Table 10, we show that LLMs can be good **zero-shot predictors** on several real-world graphs, which provides the possibility to conduct zero-shot inference on datasets without labels. Despite the effectiveness of LLMs, it still presents two problems: (1) The price of using LLMs' API is not cheap, and conducting inference on all testing nodes for large graphs incurs high costs; (2) Whether it is a locally deployed open-source LLM or a closed source LLM accessed through an API, the inference with these LLMs are much slower than GNNs, since the former has high computational resource requirements, while the latter has rate limits. One potential solution to these challenges is leveraging the knowledge of LLMs to train smaller models like GNNs, which inspires a potential application of LLMs to be used as annotators.

Based on the preliminary experimental outcomes, LLMs display encouraging results on certain datasets, thus highlighting their potential for generating high-quality pseudo-labels. However, the use of LLMs as an annotator introduces a new challenge. A key consideration lies in deciding the nodes that should be annotated. Unlike the self-labeling in GNNs[8; 34; 32], where confidence-based or information-based metrics are employed to estimate the quality of pseudo-labels. It remains a difficult task to determine the confidence of pseudo-labels generated by LLMs. Additionally, different nodes within a graph have distinct impacts on other nodes [72]. Annotating certain nodes can result in a more significant performance improvement compared to others. Consequently, the primary challenge can be summarized as follows: how can we effectively select both the critical nodes within the graph and the reliable nodes in the context of LLMs?

Taking into account the complexity of these two challenges, we don't intend to comprehensively address them in this paper. Instead, we present a preliminary study to evaluate the performance of a simple strategy: randomly selecting a subset of nodes for annotation. It is worth noting that advanced selection strategies such as active learning [72] could be adopted to improve the final performance. We leave such exploration as future work. Regarding the annotation budget, we adopt a "low labeling rate" setting, wherein we randomly select a total of 20 nodes multiplied by the number of classes. For the selected nodes, we adopt 75% of them as training nodes and the rest as validation nodes. Consequently, we annotate a total of 140 nodes in the CORA dataset and 60 nodes in the PUBMED dataset. In this part, we use GCN as the GNN model and adopt the embeddings generated by the Sentence-BERT model. The results are shown in Table 16. We can observe that training GCN on the pseudo labels can lead to satisfying performance. Particularly, it can match the performance of GCN trained on ground truth labels with 10 shots per class. As a reference, around 67% of the pseudo labels for CORA can match ground truth labels, while around 93% of the pseudo labels for PUBMED are ground truth labels.

Table 16: Performance of GCN trained on either pseudo labels generated by LLMs, or ground truth labels

	Cora	**Pubmed**
Using pseudo labels		
20 shots × #class	64.95 ± 0.98	71.70 ± 1.06
Using ground truth		
3 shots per class	52.63 ± 1.46	59.35 ± 2.67
5 shots per class	58.97 ± 1.41	65.98 ± 0.74
10 shots per class	69.87 ± 2.27	71.51 ± 0.77

Observation 17. The quality of pseudo labels is key to downstream performance.

Although we don't place significant emphasis on the selection of nodes to be labeled, the preliminary results show that there is relatively little variance among different random selections. Comparing this to the impact of pseudo labels, we observe that the quality of pseudo labels can make a significant difference. When higher quality pseudo labels are used, GNNs perform much better on PUBMED compared to CORA. This result highlights the importance of developing an approach to select confident nodes for LLMs.

Observation 18. Getting the confidence by simply

prompting the LLMs may not work since they are too "confident".
Based on previous observations, we check some simple strategies to achieve the confidence level of LLMs' outputs. Initially, we attempt to prompt the LLMs directly for their confidence level. However, we discover that most of the time, LLMs simply output a value of 1, rendering it meaningless. Examples are shown in Table 17.

Table 17: Prompts used to generate neighbor summary

Instruction
Output the confidence level in the range of 0 to 1 and the most 1 possible category of this paper as a python dict, like "prediction": "XX", "confidence": "XX"

Another potential solution is to utilize LLMs that support prediction logits, such as text-davinci-003. However, we observe that the probability of the outputs from these models is consistently close to 1, rendering the output not helpful.

5.4 Case Study: Applying LLMs to handle out-of-distribution data

Out-of-distribution (OOD) learning addresses scenarios where training and test data are drawn from different distributions. Given the ubiquity of distribution shifts in graph data [29], OOD generalization on graphs has emerged as a crucial research direction in recent years. A recent benchmark, GOOD [17], reveals that existing GNN-based models struggle with robustness when confronted with distributional shifts. In contrast, LLMs have demonstrated commendable robustness on textual data in the presence of OOD scenarios [67]. Node classification on the TAG, when disregarding graph structures, can also be considered as a text classification task. Therefore, in this section, we initiate a preliminary exploration into the application of LLMs for OOD scenarios on graphs.

Experimental Setups. We adopt the GOOD-Arxiv dataset from the GOOD benchmark [17] considering its text attribute availability. Specifically, we adopt all four types of the OOD shift: "Concept-degree", "Covariate-degree", "Concept-time", and "Covariate-time" from the GOOD. The final results are shown in Table 18. We adopt the prompt from TAPE [22] since it achieves better performance on the Ogbn-arxiv dataset. For comparison, we take the best baseline models from the GOOD benchmark.

Table 18: OOD performance comparison. "Val" means the results on the IID validation sets. "Test" indicates the results of the OOD test sets. We can see that LLMs-as-Predictors consistently outperform the best GNN-based OOD baselines. Moreover, the gap between IID performance and OOD performance is small.

	Val	Test	Best baseline (test)
concept degree	73.01	72.79	63.00
covariate degree	70.23	68.21	59.08
concept time	72.66	71.98	67.45
covariate time	74.28	74.37	71.34

Observation 19. LLMs-as-Predictors demonstrate robustness when facing OOD data.
From Table 18, we find that LLMs-as-Predictors present promising robustness against OOD data. It should be noted that we only try a simple structure-ignorant prompt, and we may further improve the OOD performance of LLMs by selecting proper in-context samples and incorporating structural information. In a nutshell, LLMs present great potential to enhance the OOD generalization capability of graph models.

6. RELATED WORK

Following our proposed two pipelines, i.e., LLMs as the Enhancers and LLMs as the Predictors, we review existing works in this section.

6.1 LLMs as the Enhancers

In the recent surge of research, increasing attention has been paid on the intersection of LLMs and GNNs in the realm of TAGs [83; 6; 78; 77; 49; 22; 86; 24; 33; 10]. Compared to shallow embeddings, LLMs can provide a richer repository of commonsense knowledge, which could potentially enhance the performance of downstream tasks [51].

Several studies employ PLMs as text encoders, transforming text attributes into node features, which can thus be classified as ***feature-level enhancement***. The integration structures vary among these works: some adopt a simple cascading structure [49; 6; 78; 37], while others opt for an iterative structure [83; 74; 77]. For those utilizing the cascading structure, preliminary investigations have been conducted to determine how the quality of text embeddings affects downstream classification performance [49]. GIANT [6] attempts to incorporate structural information into the pre-training stage of PLMs, achieving improved performance albeit with additional training overhead. SimTEG [10] suggests that using embeddings obtained through efficiently fine-tuned parameters to replace the original embeddings of pre-trained language models can solve the problem of overfitting during fine-tuning, thereby further enhancing the performance of the cascading structure. OneForAll [33] further adopts sentence embedding model to unify the feature space, and propose a unified model for diverse tasks across multiple datasets. This cascading structure has also been successfully applied to tasks such as fact verification [37] and question answering [78]. However, despite its simplicity, recent studies [83] have identified potential drawbacks of the cascading structure. Specifically, it establishes a tenuous connection between the text attribute and the graph. The embeddings generated by the PLMs do not take graph structures into account, and the parameters of the PLMs remain constant during the GNN training process. Alternatively, in the iterative structure, Graphformers [74] facilitates the co-training of PLMs and GNNs using each other's generated embeddings. GLEM [83] takes this a step further by considering pseudo labels generated by both PLMs and GNNs and incorporating them into the optimization process. DRAGON [77] successfully extends the iterative structure to the knowledge graph domain.

Compared to these studies focusing on PLMs, a recent study [22] considers the usage of embedding-invisible LLMs such as ChatGPT [45] for representation learning on TAGs, which aims to adopt LLMs to enhance the text attributes and thus can be categorized into ***text-level enhancement***. This

work introduces a prompt designed to generate explanations for the predictions made by LLMs. These generated explanations are subsequently encoded into augmented features by PLMs. Through the ensemble of these augmented features with the original features, the proposed methodology demonstrates its efficacy and accomplishes state-of-the-art performance on the OGBN-ARXIV leaderboard [23]. Nevertheless, the study offers limited analytical insights into the underlying reasons for the success of this approach. Additionally, we have identified a potential concern regarding the prompts utilized in the referenced study.

Another work pertaining to the integration of LLMs and GNNs is the Graph-Toolformer [80]. Drawing inspirations from Toolformer [56], this study utilizes LLMs as an interface to bridge the natural language commands and GNNs. This approach doesn't change the features and training of GNNs, which is out of our scope.

6.2 LLMs as the Predictors

While *LLMs-as-Enhancers* have proven to be effective, the pipeline still requires GNNs for final predictions. In a significant shift from this approach, recent studies [18; 65] have begun exploring a unique pipeline that solely relies on LLMs for final predictions. These works fall under the category of *LLMs-as-Predictors*. The first series of work focus on applying closed-source LLMs without tuning the parameters. GPT4Graph [18] evaluates the potential of LLMs in executing knowledge graph (KG) reasoning and node classification tasks. Their findings indicate that these models can deliver competitive results for short-range KG reasoning but struggle with long-range KG reasoning and node classification tasks. However, its presentation is pretty vague and they don't give the detailed format of the prompt they use. Considering the publicity of the Arxiv data, the data leakage problem in evaluation is further studied in [25]. NLGraph [65] introduces a synthetic benchmark to assess graph structure reasoning capabilities. The study primarily concentrates on traditional graph reasoning tasks such as shortest path, maximum flow, and bipartite matching, while only offering limited analysis on node classification tasks. This does not align with our central focus, primarily on graph learning, with a specific emphasis on node classification tasks. GraphText [84] further tries to apply LLMs to a broader range of non-text-attributed graphs by converting the original features into clustering centers or pseudo labels. LLM4Dyg [81] further evaluates LLMs' capability for temporal graph-related tasks. LLMGNN [4] and GPT4GNAS [66] apply LLMs-as-predictors as annotators and agents for neural architecture search, respectively.

As these closed-source LLMs only accept text-type inputs, the first type of methods requires transforming graphs into certain form of natural language, either directly using node attributes or describing the graph structure using natural language. Meanwhile, due to the input length limitations of LLMs, this transformation process often results in the loss of a considerable amount of information from the graph. Therefore, the second type of work involves fine-tuning LLMs to enable them to understand graph information represented as embeddings. InstructGLM [79] combines textual instructions with node features in embedding form, enabling LLMs to understand node features through instruction tuning. Subsequently, it predicts the type of nodes based on the given instructions. GraphGPT [62] further introduces cross-modal contrastive learning to align the graph and text feature spaces. It also introduces dual-stage instruction tuning, where the first stage adopts self-supervised instruction tuning to make LLMs better understand graph-structured information. The second stage adopts task-specific fine-tuning to allow LLMs achieve task-specific knowledge and then make predictions. GraphLLM [3] and DGTL [50] apply this pipeline to graph reasoning tasks and graph representation learning.

7. CONCLUSIONS, LIMITATIONS, AND FUTURE DIRECTIONS

In this section, we summarize our key findings, present the limitations of this study and discuss the potential directions of leveraging LLMs in graph machine learning.

7.1 Key Findings

In this paper, we propose two potential pipelines: *LLMs-as-Enhancers* and *LLMs-as-Predictors* that incorporate LLMs to handle the text-attributed graphs. Our rigorous empirical studies reveal several interesting findings which provide new insights for future studies. We highlight some key findings below and more can be found from Observation 1 to Observation 19.

Finding 1. For *LLMs-as-Enhancers*, deep sentence embedding models present effectiveness in terms of performance and efficiency. We empirically find that when we adopt deep sentence embedding models as enhancers at the feature level, they present good performance under different dataset split settings, and also scalability. This indicates that they are good candidates to enhance text attributes at the feature level.

Finding 2. For *LLMs-as-Enhancers*, the combination of LLMs' augmentations and ensembling demonstrates its effectiveness. As demonstrated in Section 4.2, when LLMs are utilized as enhancers at the text level, we observe performance improvements by ensembling the augmented attributes with the original attributes across datasets and data splits. This suggests a promising approach to enhance the performance of attribute-related tasks. The proposed pipeline involves augmenting the attributes with LLMs and subsequently ensembling the original attributes with the augmented ones.

Finding 3. For *LLMs-as-Predictors*, LLMs present preliminary effectiveness but also indicate potential evaluation problem. In Section 5, we conduct preliminary experiments on applying LLMs as predictors, utilizing both textual attributes and edge relationships. The results demonstrate that LLMs present effectiveness in processing textual attributes and achieving good zero-shot performance on certain datasets. Moreover, our analysis reveals two potential problems within the existing evaluation framework: (1) There are instances where LLMs' inaccurate predictions can also be considered reasonable, particularly in the case of citation datasets where multiple labels may be appropriate. (2) We find a potential test data leakage problem on OGBN-ARXIV, which underscores the need for a careful reconsideration of how to appropriately evaluate the performance of LLMs on real-world datasets.

7.2 Limitations

A deeper understanding of the effectiveness of text embeddings. Despite the effectiveness of deep sentence

embedding models, our understanding of why their embeddings outperform PLMs' on node classification tasks remains limited. Furthermore, we observe a performance gap between deep sentence embedding models and GLEM on the OGBN-PRODUCTS dataset, which may be related to the domains of the dataset. Moreover, as shown in Observation 4, GNNs demonstrate different levels of effectiveness on different text embeddings. However, we give limited explanations for this phenomenon. To gain a deeper understanding, we need to have a look at the original feature space and the feature space after aggregation. This phenomenon may potentially be related to the anistrophy in language model embeddings [12]. More in-depth analysis is required to better understand these phenomena.

Costs of LLM augmentations. In the work, we study TAPE and KEA to enhance the textual attributes at the text level. Although these methods have proven to be effective, they require querying LLMs' APIs at least N times for a graph with N nodes. Given the cost associated with LLMs, this poses a significant expense when dealing with large-scale datasets. Consequently, we have not presented results for the OGBN-ARXIV and OGBN-PRODUCTS datasets.

Text-formatted hand-crafted prompts to represent graphs. In Section 5, we limit our study to the use of "natural language" prompts for graph representation. However, various other formats exist for representing graphs in natural language such as XML, YAML, GML, and more [55]. Moreover, we mainly design these prompts in a hand-crafted way, which is mainly based on trial and error. It's thus worthwhile to consider exploring more prompt formats and how to come up with automatic prompts.

7.3 Future Directions

Extending the current pipelines to more tasks and more types of graphs. In this study, our primary focus is on investigating the node classification task for text-attributed graphs. Nevertheless, it remains unexplored whether these two pipelines can be extended to other graph-learning tasks or other types of graphs. Certain tasks necessitate the utilization of long-range information [11], and representing such information within LLMs' limited input context poses a significant challenge. Furthermore, we demonstrate that LLMs exhibit promising initial results in graphs containing abundant textual information, particularly in natural language. However, the exploration of their effective extension to other types of graphs with non-natural language information, such as molecular graph [13; 30], still needs further exploration.

Using LLMs more efficiently. Despite the effectiveness of LLMs, the inherent operational efficiency and operational cost of these models still pose significant challenges. Taking ChatGPT, which is accessed through an API, as an example, the current billing model incurs high costs for processing large-scale graphs. As for locally deployed open-source large models, even just using them for inference requires substantial hardware resources, not to mention training the models with parameter updates. Therefore, developing more efficient strategies to utilize LLMs is currently a challenge.

Evaluating LLMs' capability for graph learning tasks. In this paper, we briefly talk about the potential pitfalls of the current evaluation framework. There are mainly two problems: (1) the test data may already appear in the training corpus of LLMs, which is referred to as "contamination" [5] (2) the ground truth labels may present ambiguity, and the performance calculated based on them may not reflect LLMs' genuine capability. For the first problem, one possible mitigation is to use the latest dataset which is not included in the training corpus of LLMs. However, that means we need to keep collecting data and annotating them, which seems not an effective solution. For the second problem, one possible solution is to reconsider the ground truth design. For instance, for the categorization of academic papers, we may adopt a multi-label setting and select all applicable categories as the ground truth. However, for more general tasks, it remains a challenge to design more reasonable ground truths. Generally speaking, it's a valuable future direction to rethink how to properly evaluate LLMs.

Aligning the feature space of graph models and LLMs. Currently, a major obstacle hindering the wider application of LLMs in the field of graph learning is the discrepancy between the feature space of LLMs and that of graphs. This discrepancy makes it difficult for LLMs to effectively understand information in the graph domain. There are mainly two approaches to address this issue in current work. The first approach is to translate the information on the graph into natural language that LLMs can understand. The second approach involves directly inputting the graph information in the form of embeddings and then using instruction tuning to enable LLMs to understand this information. However, both methods have their evident limitations. For the first method, the translation process can result in information loss, and the inherent input length limitation of LLMs also prevents users from inputting large-scale graphs. For the second method, the introduction of tuning significantly increases computational overhead. Is there a better way to align LLMs with graphs? A recent work targeting multimodality [47] has shown new possibilities. It demonstrates that with fixed LLM parameters, only a linear transformation layer is needed to convert information from the visual domain into content that can be effectively processed by LLMs, and such an architecture also holds great potential in the field of graph machine learning.

8. REFERENCES

[1] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

[2] S. Bubeck, V. Chandrasekaran, R. Eldan, J. A. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y.-F. Li, S. M. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv*, abs/2303.12712, 2023.

[3] Z. Chai, T. Zhang, L. Wu, K. Han, X. Hu, X. Huang, and Y. Yang. Graphllm: Boosting graph reasoning ability of large language model. *arXiv preprint arXiv:2310.05845*, 2023.

[4] 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.

[5] `https://hitz-zentroa.github.io/lm-contamination/`

[5] 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. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[6] 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*, 2022.

[7] A. Creswell, M. Shanahan, and I. Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023.

[8] E. Dai, C. Aggarwal, and S. Wang. Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 227–236, New York, NY, USA, 2021. Association for Computing Machinery.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[10] K. Duan, Q. Liu, T.-S. Chua, S. Yan, W. T. Ooi, Q. Xie, and J. He. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv preprint arXiv:2308.02565*, 2023.

[11] V. P. Dwivedi, L. Rampášek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu, and D. Beaini. Long range graph benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[12] K. Ethayarajh. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[13] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric. *ArXiv*, abs/1903.02428, 2019.

[14] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *ArXiv*, abs/2303.14524, 2023.

[15] 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*, DL '98, pages 89–98, New York, NY, USA, 1998. ACM.

[16] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. *ArXiv*, abs/1704.01212, 2017.

[17] S. Gui, X. Li, L. Wang, and S. Ji. GOOD: A graph out-of-distribution benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[18] J. Guo, L. Du, and H. Liu. Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking. *arXiv preprint arXiv:2305.15066*, 2023.

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

[20] Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[21] P. He, X. Liu, J. Gao, and W. Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[22] X. He, X. Bresson, T. Laurent, and B. Hooi. Explanations as features: Llm-based features for text-attributed graphs. *arXiv preprint arXiv:2305.19523*, 2023.

[23] 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*, 33:22118–22133, 2020.

[24] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun. Gpt-gnn: Generative pre-training of graph neural networks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[25] J. Huang, X. Zhang, Q. Mei, and J. Ma. Can llms effectively leverage graph structural information: When and why. *arXiv preprint arXiv:2309.16595*, 2023.

[26] Y. Ji, Y. Gong, Y. Peng, C. Ni, P. Sun, D. Pan, B. Ma, and X. Li. Exploring chatgpt's ability to rank content: A preliminary study on consistency with human preferences. *ArXiv*, abs/2303.07610, 2023.

[27] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[28] G. Li, M. Müller, B. Ghanem, and V. Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pages 6437–6449. PMLR, 2021.

[29] H. Li, X. Wang, Z. Zhang, and W. Zhu. Out-of-distribution generalization on graphs: A survey. *arXiv preprint arXiv:2202.07987*, 2022.

[30] J. Li, Y. Liu, W. Fan, X. Wei, H. Liu, J. Tang, and Q. Li. Empowering molecule discovery for molecule-caption translation with large language models: A chatgpt perspective. *ArXiv*, abs/2306.06615, 2023.

[31] Q. Li, X. Li, L. Chen, and D. Wu. Distilling knowledge on text graph for social media attribute inference. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2024–2028, New York, NY, USA, 2022. Association for Computing Machinery.

[32] Y. Li, J. Yin, and L. Chen. Informative pseudo-labeling for graph neural networks with few labels. *Data Mining and Knowledge Discovery*, 37(1):228–254, 2023.

[33] H. Liu, J. Feng, L. Kong, N. Liang, D. Tao, Y. Chen, and M. Zhang. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149*, 2023.

[34] H. Liu, B. Hu, X. Wang, C. Shi, Z. Zhang, and J. Zhou. Confidence may cheat: Self-training on graph neural networks under distribution shift. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 1248–1258, New York, NY, USA, 2022. Association for Computing Machinery.

[35] J. Liu, C. Liu, R. Lv, K. Zhou, and Y. Zhang. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149*, 2023.

[36] W. Liu, P. Zhou, Z. Zhao, Z. Wang, Q. Ju, H. Deng, and P. Wang. K-bert: Enabling language representation with knowledge graph. In *AAAI Conference on Artificial Intelligence*, 2019.

[37] Z. Liu, C. Xiong, M. Sun, and Z. Liu. Fine-grained fact verification with kernel graph attention network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7342–7351, Online, July 2020. Association for Computational Linguistics.

[38] Y. Ma and J. Tang. *Deep Learning on Graphs*. Cambridge University Press, 2021.

[39] H. Mao, Z. Chen, W. Jin, H. Han, Y. Ma, T. Zhao, N. Shah, and J. Tang. Demystifying structural disparity in graph neural networks: Can one size fit all? *arXiv preprint arXiv:2306.01323*, 2023.

[40] A. McCallum, K. Nigam, J. D. M. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[41] A. Miaschi and F. Dell'Orletta. Contextual and non-contextual word embeddings: an in-depth linguistic investigation. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 110–119, Online, July 2020. Association for Computational Linguistics.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[43] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[44] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. A. Tezak, J. W. Kim, C. Hallacy, J. Heidecke, P. Shyam, B. Power, T. E. Nekoul, G. Sastry, G. Krueger, D. P. Schnurr, F. P. Such, K. S.-K. Hsu, M. Thompson, T. Khan, T. Sherbakov, J. Jang, P. Welinder, and L. Weng. Text and code embeddings by contrastive pre-training. *ArXiv*, abs/2201.10005, 2022.

[45] OpenAI. Introducing chatgpt, 2022.

[46] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[47] Z. Pang, Z. Xie, Y. Man, and Y.-X. Wang. Frozen transformers in language models are effective visual encoder layers. *arXiv preprint arXiv:2310.12973*, 2023.

[48] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel. Language models as knowledge bases? *ArXiv*, abs/1909.01066, 2019.

[49] S. Purchase, A. Zhao, and R. D. Mullins. Revisiting embeddings for graph neural networks. *ArXiv*, abs/2209.09338, 2022.

[50] Y. Qin, X. Wang, Z. Zhang, and W. Zhu. Disentangled representation learning with large language models for text-attributed graphs. *arXiv preprint arXiv:2310.18152*, 2023.

[51] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63:1872 – 1897, 2020.

[52] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[53] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[54] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[55] M. Roughan and S. J. Tuke. Unravelling graph-exchange file formats. *ArXiv*, abs/1503.02781, 2015.

[56] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[57] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93, Sep. 2008.

[58] C. Sun, H. Gu, and J. Hu. Scalable and adaptive graph neural networks with self-label-enhanced training. *arXiv preprint arXiv:2104.09376*, 2021.

[59] T. Sun, Y. Shao, H. Qian, X. Huang, and X. Qiu. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pages 20841–20855. PMLR, 2022.

[60] X. Sun, X. Li, J. Li, F. Wu, S. Guo, T. Zhang, and G. Wang. Text classification via large language models. *ArXiv*, abs/2305.08377, 2023.

[61] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu. Ernie: Enhanced representation through knowledge integration. *ArXiv*, abs/1904.09223, 2019.

[62] 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. *arXiv preprint arXiv:2310.13023*, 2023.

[63] 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.

[64] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[65] H. Wang, S. Feng, T. He, Z. Tan, X. Han, and Y. Tsvetkov. Can language models solve graph problems in natural language? *arXiv preprint arXiv:2305.10037*, 2023.

[66] H. Wang, Y. Gao, X. Zheng, P. Zhang, H. Chen, and J. Bu. Graph neural architecture search with gpt-4. *arXiv preprint arXiv:2310.01436*, 2023.

[67] J. Wang, X. Hu, W. Hou, H. Chen, R. Zheng, Y. Wang, L. Yang, H. Huang, W. Ye, X. Geng, et al. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *arXiv preprint arXiv:2302.12095*, 2023.

[68] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[69] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. J. Zhao, J. Li, A. Smola, and Z. Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. *ArXiv*, abs/1909.01315, 2019.

[70] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[71] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[72] Y. Wu, Y. Xu, A. Singh, Y. Yang, and A. W. Dubrawski. Active learning for graph neural networks via node feature propagation. *ArXiv*, abs/1910.07567, 2019.

[73] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu. Graph learning: A survey. *IEEE Transactions on Artificial Intelligence*, 2:109–127, 2021.

[74] J. Yang, Z. Liu, S. Xiao, C. Li, D. Lian, S. Agrawal, A. S, G. Sun, and X. Xie. Graphformers: GNN-nested transformers for representation learning on textual graph. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[75] Z. Yang, W. W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *ArXiv*, abs/1603.08861, 2016.

[76] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. *ArXiv*, abs/1809.05679, 2018.

[77] M. Yasunaga, A. Bosselut, H. Ren, X. Zhang, C. D. Manning, P. Liang, and J. Leskovec. Deep bidirectional language-knowledge graph pretraining. In *Neural Information Processing Systems (NeurIPS)*, 2022.

[78] M. Yasunaga, J. Leskovec, and P. Liang. Linkbert: Pretraining language models with document links. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8003–8016, 2022.

[79] R. Ye, C. Zhang, R. Wang, S. Xu, and Y. Zhang. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*, 2023.

[80] J. Zhang. Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt. *arXiv preprint arXiv:2304.11116*, 2023.

[81] Z. Zhang, X. Wang, Z. Zhang, H. Li, Y. Qin, S. Wu, and W. Zhu. Llm4dyg: Can large language models solve problems on dynamic graphs? *arXiv preprint arXiv:2310.17110*, 2023.

[82] Z. Zhang, A. Zhang, M. Li, and A. J. Smola. Automatic chain of thought prompting in large language models. *ArXiv*, abs/2210.03493, 2022.

[83] J. Zhao, M. Qu, C. Li, H. Yan, Q. Liu, R. Li, X. Xie, and J. Tang. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations*, 2023.

[84] J. Zhao, L. Zhuo, Y. Shen, M. Qu, K. Liu, M. Bronstein, Z. Zhu, and J. Tang. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*, 2023.

[85] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J. Nie, and J. rong Wen. A survey of large language models. *ArXiv*, abs/2303.18223, 2023.

[86] J. Zhu, Y. Cui, Y. Liu, H. Sun, X. Li, M. Pelger, L. Zhang, T. Yan, R. Zhang, and H. Zhao. Textgnn: Improving text encoder via graph neural network in sponsored search. *Proceedings of the Web Conference 2021*, 2021.

[87] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7793–7804. Curran Associates, Inc., 2020.

APPENDIX

A. DATASETS

In this work, we mainly use the following five real-world graph datasets. Their statistics are shown in Table 19.

Table 19: Statistics of the graph datasets.

Dataset	#Nodes	#Edges	Task	Metric
CORA [40]	2,708	5,429	7-class classif.	Accuracy
CITESEER * [15]	3,186	4,277	6-class classif.	Accuracy
PUBMED [57]	19,717	44,338	3-class classif.	Accuracy
OGBN-ARXIV [23]	169,343	1,166,243	40-class classif.	Accuracy
OGBN-PRODUCTS [23]	2,449,029	61,859,140	47-class classif.	Accuracy

A.1 Dataset Description

In this part, we give a brief introduction to each graph dataset. It should be noted that it's cumbersome to get the raw text attributes for some datasets, and we will elaborate them below. The structural information and label information of these datasets can be achieved from Pyg [6]. We will also release the pre-processed versions of these datasets to assist future related studies.

Cora [40] CORA is a paper citation dataset with the following seven categories: ['Rule Learning', 'Neural Networks', 'Case Based', 'Genetic Algorithms', 'Theory', 'Reinforcement Learning', 'Probabilistic Methods']. The raw text attributes can be obtained from `https://people.cs.umass.edu/~mccallum/data.html`.

Citeseer [15] CITESEER is a paper citation dataset with the following seven categories: ["Agents", "ML", "IR", "DB", "HCI", "AI"]. Note that we find that the TAG versopm only contains the text attributes for 3186 nodes. As a result, we take the graph consisted of these 3186 nodes with 4277 edges.

Pubmed [57] PUBMED is a paper citation dataset consisting scientific journals collected from the PubMed database with the following three categories: ['Diabetes Mellitus, Experimental', 'Diabetes Mellitus Type 1', 'Diabetes Mellitus Type 2'].

Ogbn-arxiv and Ogbn-products [23] These dataset are selected from the popular OGB benchmark [23], and descriptions for these datasets can be found in `https://ogb.stanford.edu/docs/nodeprop`.

B. EXPERIMENT SETUPS

B.1 Computing Environment

We implement all the baseline models with PyG [13], DGL [69], and transformers [71] modules. The experiments were conducted in a GPU server with eight NVIDIA RTX A5000 GPUs, each with 24GB VRAM.

B.2 Hyperparameters

For RevGAT, GraphSage, and SAGN models, we directly adopt the best hyperparameters from the OGB leaderboard [7]. For Deberta-base on CORA and PUBMED, we follow the hyperparameter setting of TAPE [22]. In terms of GLEM, for the LM part, we follow the hyperparameter setting in their reporsitory [8]. For GCN, GAT, MLP, we use the following hyperparameter search range.

(a) **Hidden dimension:** {8, 16, 32, 64, 128, 256}.
(b) **Number of layers:** {1, 2, 3}
(c) **Normalization:** {None, BatchNorm};
(d) **Learning rate:** {1e-2, 5e-2, 5e-3, 1e-3}
(e) **Weight Decay:** {1e-5, 5e-5, 5e-4, 0}
(f) **Dropout:** {0., 0.1, 0.5, 0.8}
(g) **Number of heads for GAT:** {1, 4, 8}

C. DEMONSTRATIONS OF TAPE

Examples for Pubmed After analyzing the PUBMED dataset, we find an interesting phenomenon that sometimes the label of the paper just appears in the raw text attributes. An example is shown in Table 20. This property of PUBMED may be related to the superior zero-shot performance of LLMs on this dataset. This can also potentially explain why GCN and GAT are outperformed by MLP in the high labeling ratio. When the link between node attributes and node labels can be easily found and adequate to determine the categories, incorporating neighbors coming from other categories will introduce noise.

Table 20: An illustrative example for PUBMED

Title: Predictive power of sequential measures of albuminuria for progression to ESRD or death in Pima Indians with **type 2 diabetes**.
... (content omitted here)
Ground truth label: Diabetes Mellitus Type 2

[6] `https://pytorch-geometric.readthedocs.io/en/latest/modules/data.html`

[7] `https://github.com/snap-stanford/ogb`

[8] `https://github.com/AndyJZhao/GLEM`