Distilling Large Language Models for Text-Attributed Graph Learning

Bo Pan
Emory University
Department of Computer Science
Atlanta, GA, USA
bo.pan@emory.edu

Zheng Zhang
Emory University
Department of Computer Science
Atlanta, GA, USA
zheng.zhang@emory.edu

Yifei Zhang
Emory University
Department of Computer Science
Atlanta, GA, USA
yifei.zhang2@emory.edu

Yuntong Hu
Emory University
Department of Computer Science
Atlanta, GA, USA
yuntong.hu@emory.edu

Liang Zhao
Emory University
Department of Computer Science
Atlanta, GA, USA
liang.zhao@emory.edu

ABSTRACT

Text-Attributed Graphs (TAGs) are graphs of connected textual documents. Graph models can efficiently learn TAGs, but their training heavily relies on human-annotated labels, which are scarce or even unavailable in many applications. Large language models (LLMs) have recently demonstrated remarkable capabilities in few-shot and zero-shot TAG learning, but they suffer from scalability, cost, and privacy issues. Therefore, in this work, we focus on synergizing LLMs and graph models with their complementary strengths by distilling the power of LLMs into a local graph model on TAG learning. To address the inherent gaps between LLMs (generative models for texts) and graph models (discriminative models for graphs), we propose first to let LLMs teach an interpreter with rich rationale and then let a student model mimic the interpreter's reasoning without LLMs' rationale. We convert LLM's textual rationales to multi-level graph rationales to train the interpreter model and align the student model with the interpreter model based on the features of TAGs. Extensive experiments validate the efficacy of our proposed framework.

CCS CONCEPTS

• **Computing methodologies → Machine learning**.

KEYWORDS

Large language models; knowledge distillation; text-attributed graphs

1 INTRODUCTION

Text-attributed graphs (TAGs) are a type of graph where each node is associated with a text entity, such as a document, and edges reflect the relationships between these nodes. TAGs harness the power of containing both semantic content and structural relations and thus have been predominantly utilized across various domains, including citation networks, e-commerce networks, social media, recommendation systems, and web page analytics, etc. [44, 45] The exploration of TAGs has been attracting significant interest due to its potential to transcend the conventional analysis of independent and identically distributed (i.i.d.) text features and focus on the relationship of text features. Current research on TAG learning typically adopts the pipeline of first extracting the text representations with a language model (LM), then feeding extracted text representation into a Graph Neural Network (GNN) to extract node embeddings with structural information [5, 8, 44, 45]. Such a pipeline allows the simultaneous capture of semantic and structural insights, yielding effective learning on TAGs. However, the training of GNN typically heavily relies on annotated labels, which are tedious to prepare and may not be available in numerous tasks in TAGs [35, 39].

The recent emergence of Large Language Models (LLMs) brought light to solving the difficulty of data scarcity. For TAG learning, LLMs have proven zero-shot capabilities [3, 4], and even become new state-of-the-art on some datasets [19]. Despite the promise, the deployment, fine-tuning, and maintenance of LLMs require excessive resources, which may not be afforded by most of the institutions whose devices are not powerful enough, largely limiting their applicability. The cost of using public LLM APIs, such as Chat-GPT, can be huge, especially for the TAG problem, which requires subgraphs of documents as the inputs [34]. The practical applicability is further deteriorated by the privacy concerns of transferring sensitive data (e.g., in the network of health, social media, and finance, etc.) to public LLMs APIs [37]. These issues—scalability, cost, and privacy—underscore the necessity for a localized graph model that retains the advanced capabilities of LLMs without their associated drawbacks.

Given the need to have a localized model that also enjoys LLMs' power on TAG learning, a straightforward idea is knowledge distillation. For **language** models, previous research succeeded in

distilling LLMs into smaller models and achieved comparable performance on specific tasks. Recent research [15, 16, 22] found that jointly distilling the answers and rationales can be more effective than only using answers for distillation. However, the distillation of LLMs becomes a non-trivial task when the student models become **graph** models, especially when consider leveraging the rationales, due to the following challenges: 1) *How to let language models teach graph models?* Language models are "eloquent" teachers, which are typically generative models that output expressive information; while graph models, such as graph neural networks, have very succinct inputs and outputs (e.g., class labels), which may limit the knowledge absorption if using ordinary knowledge distillation by aligning the outputs. It is challenging yet important to make graph models sufficiently absorb expressive knowledge during training, while still can seamlessly adapt to succinct input and output when predicting. 2) *How to transfer text rationales to graph rationales?* The rationales provided by LLMs are textual data that use natural language to explain the reasoning process, while graph rationale focuses on the salient areas in graphs most important to prediction. Seamlessly transferring across these heterogeneous rationales is seriously under-explored and challenging. 3) *How to synergize text and graph information during knowledge distillation?* Text-attributed graphs encompass the synergy of textual and graph topology information which are highly heterogeneous. How to ensure that both of these two types of knowledge as well as their interplay can be well preserved after knowledge distillation is a difficult and open problem.

To tackle these challenges, we propose a novel framework to distill LLMs into graph models. Our approach leverages the expressive outputs of LLMs during training through a two-stage distillation process. First, instead of having LLMs directly teach the student graph model, we introduce an *interpreter* model that has a similar structure to the student model but can take the expressive outputs of LLMs as input. The student model is then aligned with the interpreter model that infers using raw data without rationales, enabling it to infer when LLMs are not available at test time. To ensure the interpreter model is well-trained, we convert the LLM's textual rationales into enhanced features at the text-level, structure-level, and message-level, which inform its predictions. For effective alignment between the student and interpreter models, we propose a new TAG model alignment method that considers the feature discrepancies between text and graph embeddings. This approach ensures that the student model inherits the knowledge from the interpreter and can perform inference without relying on the expressive rationales from LLMs.

Our contributions can be summarized as follows:

- We propose a new framework for distilling LLMs' knowledge to graph models for TAG learning. Our proposed framework achieves such knowledge distillation by letting LLMs output rationales to train an interpreter model, which is then aligned with a student model with no dependency on LLMs.
- We propose to convert textual rationales to text-level, structure-level, and message-level graph rationales as enhanced features for the interpreter model, and LLM-generated pseudo-labels and pseudo-soft labels as supervision to train the interpreter model.

- We propose a semantics and structure-aware TAG model aligning method. The proposed alignment method preserves the text and graph information in aligning TAG models, thus allowing the student model to better align with the interpreter model.
- We conducted comprehensive experiments to validate the performance of our proposed framework. The proposed method consistently beat the baseline methods by an average improvement of 6.2% across four datasets.

2 RELATED WORK

2.1 LLMs for Text-Attributed Graphs Learning

Current research on applying LLMs to help TAG learning contains two categories: LLM-as-predictor and LLM-as-enhancer. The first stream of methods starts with the research of directly utilizing LLMs for zero-shot prediction [3, 4]. Research shows LLMs even become new state-of-the-art on some datasets [18, 19]. A new focus of recent research is tuning LLMs for better prediction on graphs [2, 31, 38, 43]. The second stream of work, LLM-as-enhancer, focuses on using LLMs to enhance the features of TAG. For example, LLM has been proven the power of node feature enhancement [13], edge editing [30], etc. Such power can greatly benefit the learning on TAGs by providing augmented data and features for training the models [13, 30]. Although LLMs have superior power of text feature understanding or local link modification, they still lack the ability of capturing higher-hop neighbor information due to the constraints on input text length, and such ability is the advantage of GNNs. Therefore, using LLMs to enhance the features and combining them with GNNs has become a new state-of-the-art paradigm in TAG learning [4, 13, 30]. However, both these two streams of work requires calling LLMs during the test time, which has the issues of cost and privacy. How to benefit from LLMs in the training but predict without LLMs during test time is still an open problem.

2.2 Knowledge Distillation of LLMs

Previous research in the NLP domain succeeded in distilling larger-scale pre-trained language models into smaller models and achieving comparable performance [6, 20, 27]. In the LLM's era, recent work found distilling the rationales as well as answers generated by LLMs in a multi-task learning setting can significantly improve the performance of the student model [1, 15, 16, 23, 25, 29, 42]. Specifically, when distilling LLMs to smaller language models, the rationales like explanations [16] or chain-of-thought [29] are also used as supervision for the student model to better learn the reasoning capabilities. When the target models are graph models, one work attempted to use LLM as an annotator for node classification [4], but it only considered distilling the predictions and failed to leverage the rationales of LLMs. To the best of our knowledge, no existing work has attempted to leverage the rationales to help the knowledge distillation into graph models.

2.3 Privileged Information

Privileged Information [32] introduces a teacher who provides additional information to a student model during the learning process. The underlying idea is that the teacher's extra explanations can help the student develop a more effective model. *Generalized distillation*

[24] is a framework that unifies privileged information and knowledge distillation [14, 41]. It gives a more general form of knowledge distillation that allows the teacher model to have privileged input information than the student model. The generalized distillation framework first learns a teacher model with the data where the input is privileged information and generates a set of soft labels for each data. Then the student model is learned with the original features by aligning the soft labels with the teacher model.

3 PRELIMINARIES

Learning on Text-Attributed Graphs. TAG learning aims to learn representations of graphs, each of whose nodes is associated with a text feature. Most existing work on TAG learning follows the paradigm of first using a pre-trained language model (LM) to encode the text features into embeddings, then using a graph neural network (GNN) to aggregate the neighbor information to get the node embedding. This approach leverages the strengths of both language models in understanding textual information and GNNs in capturing the relational structure of the graph. In this work, we also follow this paradigm.

Formally, a TAG can be represented as $\mathcal{G} = (\mathcal{V}, A, \mathcal{X})$, where $\mathcal{V} = \{v_0, v_1, ..., v_{N-1}\}$ is a set of N nodes, $A \in \{0, 1\}^{N \times N}$ is the adjacency matrix, and $\mathcal{X} = \{x_0, x_1, ..., x_{N-1}\}$ is the set of text features where x_n is the text feature associated with node $v_n \in \mathcal{V}$. For a node v_n, its initial text embedding is obtained by

$$h_{n,0} = \text{LM}(x_n) \tag{1}$$

where LM is the text encoder, usually implemented with a BERT-like language model, and x_n is the text associated with node v_n. After obtaining the text embeddings as initial node embeddings, a GNN adopts message passing to learn the structure-aware embeddings for each node. A general message-passing operation can be expressed as:

$$h_{n,k} = \text{UPD}\Big(h_{n,k-1}, \\ \text{AGG}\Big(\{\text{MSG}(h_{n,k-1}, h_{m,k-1})\}_{v_m \in \mathcal{N}(v_n)}\Big)\Big) \tag{2}$$

where $h_{n,k}$ represents the embedding of node v_n in the $k-$th layer of the graph neural network, $\mathcal{N}(v_n)$ is the set of neighbor nodes of v_n. The AGG function is used for aggregating the neighbor node embeddings and UPD is the function to update the embedding of v_n based on the aggregated neighbor node embedding. Finally, the node classification is made with the final node representation by $\hat{t}_n = MLP(h_{n,l})$ where l is the number of message passing layers and $\hat{t}_n$ is the predicted soft labels (logits).

4 METHODOLOGIES

4.1 Problem Formulation

In this work, we delve into the task of distilling an LLM to a local model for TAGs. We focus on the foundational task of node classification. Given a text-attributed graph $\mathcal{G} = (\mathcal{V}, A, \mathcal{X})$ and a large language model LLM, the goal is to train a student model, which consists of a language model $LM^{\mathcal{S}}$ and a graph neural network model $GNN^{\mathcal{S}}$ with the capability to infer using the features $\mathcal{V}, A, \mathcal{X}$. During the learning process, no ground truth labels are provided, and only a subset of nodes $\mathcal{V}_{train}$ and their textual features $\mathcal{X}_{train}$ are allowed to be exposed to LLMs.

This problem presents several unique challenges, especially when considering leveraging LLM's rationales in the distillation, due to the following reasons:

- **Difficulty in leveraging language models to teach graph models.** The teacher model (LLM) and the student model (graph models) differ significantly in their model architectures and output forms. LLMs generate succinct text information, while the output of the student graph models are merely class labels. Traditional knowledge distillation by aligning the logits and parameters cannot make the student model fully absorb the knowledge from the teacher model.
- **Difficulty in transferring text rationales to graph rationales.** LLMs can provide textual rationales for TAG learning, which the graph models cannot naturally understand. Transferring the text rationales to graph rationales is a challenging yet under-explored problem.
- **Difficulty in synergizing text and graph information in model distillation.** In the distillation process, the student model should be well aligned with the teacher model for a better understanding of both the text and structure information. It is vital but challenging to preserve text and graph knowledge and their interplay in the distillation.

4.2 A General Framework for Distilling LLM to Graph Models

To tackle the difficulty in leveraging language models to teach graph models, we propose a framework that bridges the gap between these two types of models by enabling the student graph model to comprehend and apply the knowledge derived from LLMs.

LLMs can provide rich knowledge on TAG learning, i.e., rationales. The rationales can be depicted as the important nodes, edges, and text features for each prediction. Graph models, which are discriminative models, can benefit from the rationales by taking them as input. Taking the rationales as input helps the model make predictions easier since the information in the rationale is exactly leading to the prediction. To make use of LLM's knowledge, a naive idea is to use LLM-provided rich knowledge as additional features to train the graph model (Fig. 2 (a)). However, although the rationales can benefit the training process, using them as inputs to train the model leads to even worse test performance due to the distribution shift of features at the test time [36]. An alternative solution is to only leverage succinct knowledge (i.e. LLM's predictions) as supervision to train the model so that the training and test features can be in the same distribution (Fig. 2 (b)). However, this method falls short of leveraging the full rationale power of LLMs, which encompasses rich information crucial for the student model's understanding of the predictions.

To address this problem and ensure the model can generalize to original features without rationales at test time, we propose a solution which is first to use LLMs' rationale to train an intermediate model, then use the intermediate model to train the student model with the same architecture but takes raw features without rationales (Fig. 2 (c)).

Leveraging this idea in knowledge distillation, we propose a novel framework for the LLM-to-graph model knowledge distillation problem, illustrated in Fig. 1 (a). Our proposed framework
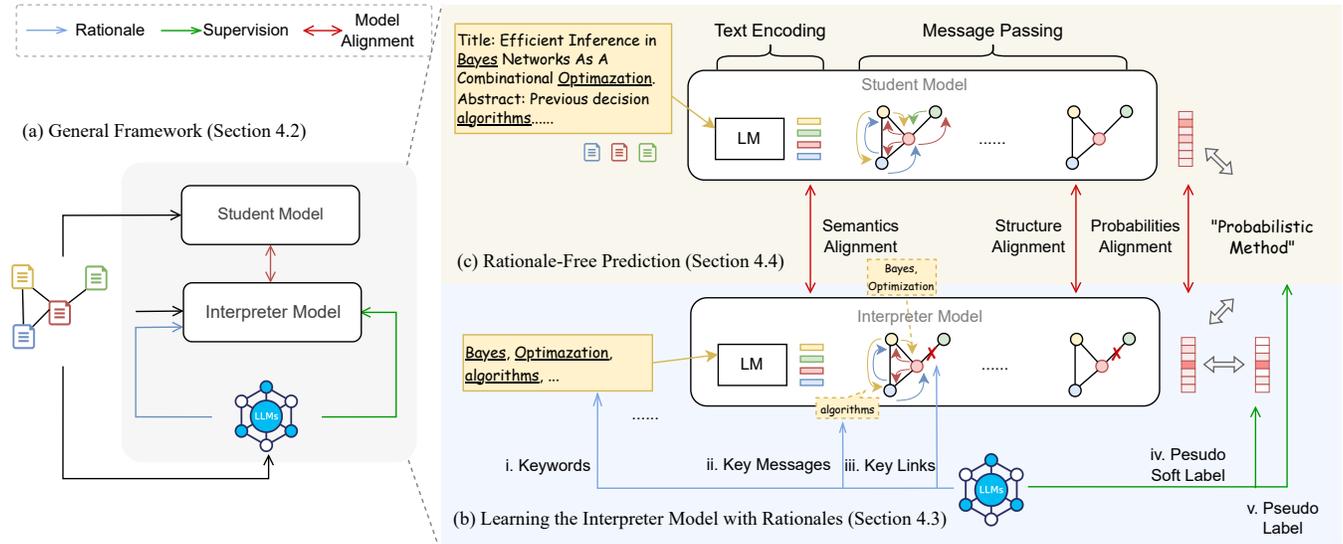
Figure 1: Illustration of our proposed LLM to graph model knowledge distillation framework. (a) The general distillation framework. We propose to distill the knowledge of an LLM by leveraging the LLM-generated rationales and supervision to train an interpreter model, then align the student model on raw features with the interpreter model. (b) The training of the interpreter model with rationales and pseudo-supervision. The interpreter model takes rationales as input, including keywords, key edges, and key messages. LLM-generated pseudo supervision, including pseudo-label and soft labels, is used to train the interpreter model. (c) The proposed model alignment framework. The student model which takes original features as input is aligned with the interpreter model on text and graph levels based on the discrepancy between raw inputs and rationale-enhanced inputs.
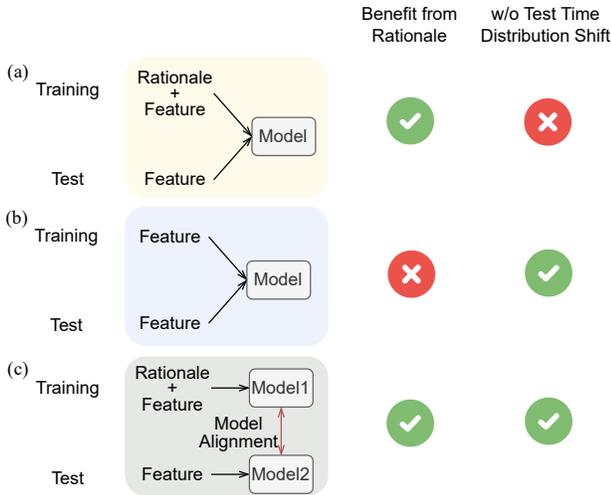

Figure 2: Different approaches to incorporate rationales to train graph models.

involves an intermediate *Interpreter* model, which bridges the LLM and the student model. Specifically, to extract the knowledge of LLMs on TAG learnings, we let LLMs provide rationales (the blue arrow) to enhance the features and provide supervision (the green arrow) to train the model. Since the interpreter model and the student model share a similar model structure, the student model can be aligned with the interpreter model by aligning their latent embeddings (the red arrow).

Given this general framework, several specific questions arise that need to be addressed to enhance its practical application. These include: (1) how to incorporate LLM-generated textual rationales into the interpreter graph model, and (2) how to effectively align the student model with the interpreter model. The remainder of this section addresses these questions, with the first question discussed in Section 4.3 and the second in Section 4.4.

4.3 Interpreter Model: Zero-Shot TAG Learning with Rationales

To tackle the difficulty in transferring text rationales to graph rationales, in this section we propose a new method to convert TAG rationales into multiple-level enhanced features. In this part, the goal is to train a strong interpreter model, consisting of $LM^{\mathcal{T}}$ and $GNN^{\mathcal{T}}$. The rationales behind a TAG model decision can be depicted as:

A prediction is primarily made based on which part(s) of text on which neighbor node(s).

Since the graph models cannot directly take the textual rationales, to bridge the style of textual and graph rationales, we convert the textual rationale into three forms of feature enhancements that help with the prediction of the answer: i) keywords in texts, ii) key links around the central node, and iii) key semantic messages from the neighbors. To operationalize the training, we leverage LLMs

to generate supervision to train the interpreter model, including iv) pseudo-soft labels and v) pseudo-labels. The training of the interpreter model with rationales is illustrated in Fig. 1 (b).

We prompt such prediction and reasoning in a three-step manner: 1) We input the text attributes of each node, along with those of its neighbors, into LLMs to generate pseudo-labels and pseudo-soft labels. 2) We then supply the node's text and its pseudo-label to LLMs to identify critical keywords within the node's attributes for the classification. 3) We feed the text of the node and its neighbors along with its pseudo-label into LLMs to identify essential links and messages for the classification. Each of these steps will be introduced in detail in the following.

Pseudo-Label and Soft Label Creation. Since soft labels can have more information compared to hard categories in knowledge distillation [24], we first leverage the zero-shot abilities of LLMs to generate the pseudo-labels and pseudo-soft labels to assist in the training of the interpreter model. The generated labels are also used as the targeted answers for generating rationales. The process of label and soft-label creation can be written as

$$l_n, y_n = LLM(x_n, \{x_i\}_{\{i|v_i \in \mathcal{N}(v_n)\}}; prompt_L) \quad (3)$$

where y_n and l_n are the LLM-predicted pseudo-label and soft label for node v_n, respectively. $LLM(x_n, \{x_i\}_{\{i|v_i \in \mathcal{N}(v_n)\}}; prompt_L)$ means calling LLM with the prompt $prompt_L$ and variables x_n and $\{x_i\}_{\{i|v_i \in \mathcal{N}(v_n)\}}$. A simplified example of $prompt_L$ is given below for easier understanding. Note that the format conversion from textual outputs to numerical values is omitted in all the mathematical expressions.[1]

> **A simplified example of $prompt_L$**
>
> **Input:** We want to classify a paper into the following categories: [Neural Networks, Generic Algorithms, ...]. Please identify logits-like probabilities for each class and give your final classification. The paper is ... Its neighbors are ...
> **Response:** {Probabilities: [0.05, 0.85, ...], Category: 'Generic Algorithm'}

Keyword Recognition. LLMs have been proven to have the capability to extract keywords in languages [10]. In the interpreter model, we leverage the reasoning power of LLMs to extract the important keywords that are most helpful for the classification of the text to the predicted class, thus removing the words that can mislead the classification. We then concatenate the keywords and feed them to the local LM to extract text embeddings. Formally, assuming $prompt_{KW}$ is the prompt for keyword recognition, for a node v_n with a text attribute x_n, we feed it to an LLM with $prompt_{KW}$ to obtain the recognized keywords. The enhanced text embedding can be obtained as

$$h_{n,0}^{\mathcal{T}} = LM^{\mathcal{T}} \left([LLM(x_n, y_n; prompt_{KW})]\right) \quad (4)$$

where $LLM(x_n, y_n; prompt_{KW})$ means utilizing LLMs to extract a list of important keywords of x_n that helps to classify it to y_n with the prompt $prompt_{KW}$. Here the output of the function LLM with $prompt_{KW}$ is seen as the set of keywords. $[\cdot]$ denotes concatenating

---

[1]All the formatting instructions in prompts are omitted.

the keywords with spaces as the separator. A simplified example of $prompt_{KW}$ is given below.

> **A simplified example of $prompt_{KW}$**
>
> **Input:** We want to classify a paper into the following categories: [Neural Networks, Generic Algorithms, ...]. Please identify at most 5 words in the provided text that help most with the classification to 'Neural Networks'. The paper is ... Its neighbors are ...
> **Response:** [Algorithm, Optimization, Bayesian]

Key Links and Messages Recognition. For structural reasoning, we prompt LLM to identify the key links around a specific central node by identifying a subset of neighbors (key links), and also the keywords in each of the neighbor nodes (key messages), that are important for the central node's classification to the predicted class. The identified key links are used as edited local structures in message passing, and the key messages are used to replace the messages in the first graph convolutional layer. Formally, for a central node v_n, we provide LLM with its text attribute x_n, its neighbors' texts $\{x_i\}_{\{i|v_i \in \mathcal{N}(v_n)\}}$, the pseudo-label y_n and the prompt for this task $prompt_{KL}$, and extract the important neighbor set and keyword set from LLM's response. For expression simplicity, we write them as the function LLM's two outputs with $prompt_{KL}$. The key links and message recognition are written as

$$\mathcal{N}'(v_n), \{w_{n,i}\}_{\{i|v_i \in \mathcal{N}'(v_n)\}}$$
$$= LLM\left(x_n, \{x_i\}_{\{i|v_i \in \mathcal{N}(v_n)\}}, y_n; prompt_{KL}\right) \quad (5)$$

where $\mathcal{N}'(v_n)$ is the key neighbor nodes and $w_{n,i}$ is the keywords in the message from node v_i to the central node v_n. With the enhanced structural features, the message passing is done by

$$h_{n,k}^{\mathcal{T}} = \text{UPD}(h_{n,k-1}^{\mathcal{T}},$$
$$\text{AGG}(\{\text{MSG}(h_{n,k-1}^{\mathcal{T}}, h_{m,k-1}^{\mathcal{T}})|v_m \in \mathcal{N}'(v_n)\})) \quad (6)$$

The messages in the first graph convolutional layer are replaced with

$$\text{MSG}(h_{n,0}^{\mathcal{T}}, h_{i,0}^{\mathcal{T}}) = LM^{\mathcal{T}}\left([w_{n,i}]\right) \quad (7)$$

where $[\cdot]$ denotes concatenating the words in the key message from v_i to v_n. A simplified example of $prompt_{KL}$ is given as below.

> **A simplified example of $prompt_{KL}$**
>
> **Input:** We want to classify a paper in a citation network to the following categories: [Neural Networks, Generic Algorithms, ...] Please identify a subset of important neighbors and at most 5 keywords of each important neighbor that help most to classify the central node into the category of 'Neural Networks'. The paper is ... Its neighbors are ...
> **Response:** {Node 0: ['Bayesian', 'Learning'], Node 248: ['Optimize', 'Convex']}

Training Objective of the Interpreter Model. With the above LLM-provided rationales, the learning of TAG is achieved with Eq. 4 ($LM^{\mathcal{T}}$) and Eq. 6 ($GNN^{\mathcal{T}}$). After obtaining the last layer's node embedding $h_{n,l}^{\mathcal{T}}$ with Eq. 6, the prediction of the interpreter model is made by $\hat{t}_n^{\mathcal{T}} = MLP^{\mathcal{T}}(h_{n,l}^{\mathcal{T}})$, where $\hat{t}_n^{\mathcal{T}}$ is the predicted soft

label. The interpreter model is trained to predict the label, and soft labels in a multi-task learning manner. The overall loss function to train the interpreter model is given by:

$$l^{\mathcal{T}} = \sum_{\{n\}_{v_n \in \mathcal{V}_{train}}} \left(l_{label}^{\mathcal{T}}(\hat{t}_n^{\mathcal{T}}, y_n) + \lambda_1 l_{logits}^{\mathcal{T}}(\hat{t}_n^{\mathcal{T}}, l_n) \right) \quad (8)$$

where l_{label} is implemented with the Cross-Entropy (CE) loss, l_{logits} is implemented with the Mean Square Error (MSE) loss, and λ_1 is a hyper-parameter.

4.4 Semantics and Structure-Aware TAG Model Alignment

To achieve better LLM-free prediction at test time, we propose a new model alignment method for TAGs, as illustrated in Fig. 1 (c). The proposed method jointly considers the textual embeddings (LM-extracted) and structural embeddings (GNN-extracted) between the interpreter and student models, thus allowing the student model to give more similar predictions to the interpreter model but does not need to take rationales as input.

Semantics Alignment. For semantic representation, we extract text embeddings from the interpreter model and student model with weights considering their occurrence frequency in the graph structures. We also focus more on those nodes whose LLM-extracted keywords are more different from their raw text when aligning them. Formally, the semantic alignment loss can be written as:

$$l_{semantic}^{\mathcal{S}} = \sum_{\{n\}_{v_n \in \mathcal{V}_{train}}} \frac{\text{Deg}(v_n)}{\text{sim}_t(x_n, x_n^{\mathcal{T}})} \cdot d(h_{n,0}^{\mathcal{S}}, h_{n,0}^{\mathcal{T}}) \quad (9)$$

where $\text{Deg}(v_n)$ is the degree of v_n, $\text{sim}_t(x_n, x_n^{\mathcal{T}})$ is the semantic similarity between the raw text feature and the LLM-enhanced text feature of node v_n, which can be calculated with the cosine similarity of their embeddings extracted from a pre-trained language model like Bert, and $d(h_{n,0}^{\mathcal{S}}, h_{n,0}^{\mathcal{T}})$ is the distance between their text embeddings that we aim to minimize.

Structure Alignment. In structural alignment, we also focus more on those nodes with more discrepancy between the original structure and the rationale-enhanced structure. Since in our proposed method, the structure enhancement is done by selecting important neighbors from all neighbors, the discrepancy can be calculated as the difference of neighbor node numbers. Formally, the structural alignment loss can be written as:

$$l_{structural}^{\mathcal{S}} = \sum_{\{n\}_{v_n \in \mathcal{V}_{train}}} \frac{\text{Deg}(v_n)}{\text{sim}_s(\mathcal{N}(v_n), \mathcal{N}'(v_n))} \cdot d(h_{n,l}^{\mathcal{S}}, h_{n,l}^{\mathcal{T}}) \quad (10)$$

where $\text{sim}_s(\mathcal{N}(v_n), \mathcal{N}'(v_n))$ measures the similarity between v_n's original neighbor structure and its enhanced neighbor structure, calculated by

$$\text{sim}_s(\mathcal{N}(v_n), \mathcal{N}'(v_n)) = \frac{1}{|\mathcal{N}_k(v_n)| - |\mathcal{N}'_k(v_n)|} \quad (11)$$

and $d(h_{n,l}^{\mathcal{S}}, h_{n,l}^{\mathcal{T}})$ is a measure of the distance of node embeddings, which we aim to minimize.

Training Objective of Model Alignment. In addition to the above two distillation loss functions, we also align the logits between the interpreter and the student model. The overall objective

in model alignment can be written as:

$$l^{\mathcal{S}} = \underbrace{l_{label}^{\mathcal{S}} + \lambda_2 l_{logits}^{\mathcal{S}}}_{\text{standard distillation}} + \underbrace{\lambda_3 l_{semantic}^{\mathcal{S}} + \lambda_4 l_{structural}^{\mathcal{S}}}_{\text{TAG model alignment}} \quad (12)$$

where $l_{label}^{\mathcal{S}}$ is the prediction loss of the student model, which is usually calculated with cross-entropy loss. $l_{logits}^{\mathcal{S}}$ is the logits alignment loss between the interpreter and student model, which is usually implemented with the mean square error (MSE) loss. $\lambda_2, \lambda_3, \lambda_4$ are the hyper-parameters.

5 EXPERIMENTS

5.1 Experimental Setting

Datasets. We evaluate our proposed framework on four text-attributed graph datasets: Cora [26], Pubmed [28], ogbn-products [17] and arxiv-2023 [13]. Each dataset is described in detail as follows: **Cora** is a paper citation network dataset consisting of 2,708 scientific publications from the computer science domain. The shallow node attributes are represented by a 1,433-dimensional binary vector indicating the presence of specific words in the document. **Pubmed** is a paper citation dataset consisting of 19,717 scientific journals collected from the PubMed database, which focuses on the biomedical domain. Each publication is classified into one of three categories related to diabetes. The node attributes are represented by a Term Frequency-Inverse Document Frequency (TF-IDF) weighted word vector with 500 unique words. **Ogbn-products** [17] is an Amazon co-purchase network where each node represents a product. The corresponding input raw text consists of titles and descriptions of products. The node labels are categories of products, and there are 47 distinct categories. We use the same subset of ogbn-products as in [13]. Shallow embeddings are extracted with BoW. **Arxiv-2023** [13] is a dataset comprising papers published in 2023 or later. This dataset is specifically designed to address the concern of potential label leakage, as it includes papers that are beyond the knowledge cutoff for models like GPT-3.5. This dataset helps in evaluating the model's performance on more recent and unseen data. The shallow embeddings are extracted with word2vec. The statistics of each dataset are given in Table 2. Following previous work [13], for all datasets, we randomly split the training/validation/test set with the ratio of 60/20/20.

Table 2: Statistics of the TAG datasets.

Dataset	#Nodes	#Edges	#Classes
Cora	2,708	5,429	7
Pubmed	19,717	44,338	3
ogbn-products	54,025	74,420	47
arxiv-2023	46,198	78,548	40

Comparison Methods. Since our problem setting is LLM-free prediction, we only compare to methods that is independent with LLMs at test time. For a fair comparison, we compare our proposed method with distilling the prediction without rationales (**LLM as Annotator**) to train the local models like LMs and GNNs. Specifically, our comparison methods include:

Table 1: Main Results of Zero-Shot Test-Time LLM-Free Node Classification.

LLM's Role	Method	Backbone	Cora	PubMed	ogbn-products	arxiv-2023
LLM as Annotator	LM	Bert	0.7400±0.0175	0.9058±0.0046	0.7020±0.0033	0.6840±0.0122
		DistilBert	0.7355±0.0163	0.9028±0.0053	0.7080±0.0040	0.6821±0.0100
		Deberta	0.7385±0.0127	0.9020±0.0057	0.7074±0.0056	0.6789±0.0185
	GNN (Shallow)	GCN	0.7126±0.0213	0.8322±0.0076	0.5593±0.0031	0.6355±0.0032
		GAT	0.7186±0.0346	0.8122±0.0282	0.5583±0.0010	0.6351±0.0060
		SAGE	0.7149±0.0223	0.8287±0.0107	0.5475±0.0023	0.6460±0.0006
	GNN (PLM)	GCN	0.6720±0.0333	0.8136±0.0099	0.6944±0.0111	0.6647±0.0068
		GAT	0.6628±0.0434	0.7996±0.0229	0.7049±0.0043	0.6675±0.0059
		SAGE	0.6619±0.0191	0.7968±0.0118	0.6879±0.0067	0.6748±0.0067
	GIANT	GCN	0.7205±0.0045	0.8122±0.0048	0.6977±0.0042	0.6679±0.0067
		GAT	0.7233±0.0024	0.8077±0.0079	0.7189±0.0030	0.6822±0.0073
		SAGE	0.7145±0.0033	0.8202±0.0046	0.6869±0.0119	0.6683±0.0037
	GLEM	RevGAT	0.7312±0.0017	0.7863±0.0122	0.7126±0.0059	0.6823±0.0030
	SIMTEG	SAGE	0.7327±0.0020	0.8327±0.0113	0.7037±0.0053	0.7233±0.0034
LLM as Teacher	**(Proposed)**	GCN	0.8237±0.0187	0.9215±0.0096	0.7333±0.0025	0.7801±0.0424
		GAT	0.8237±0.0137	0.9189±0.0019	0.7346±0.0030	0.7838±0.0424
		SAGE	0.8210±0.0296	0.9217±0.0105	0.7283±0.0015	0.7918±0.0456

(1) **LMs** (Language Models): Pre-trained language models fine-tuned by LLM-provided pseudo-labels are adopted as the first type of comparison method. Pre-trained Bert [7], DistilBert [27] and DeBERTa [12] are tested as model backbones.

(2) **GNNs**: For GNNs, we tested GCN [21], GAT [33] and Graph-SAGE [11] as backbones. We compare our method with shallow and PLM initial node features:

　(a) **Shallow**: Text embeddings provided by the PyG library [9], extracted with shallow methods including BoW, TD-IDF and word2vec, as introduced in each dataset's description.

　(b) **PLM**: Text embeddings extracted from Bert-like Pre-trained Language Models are used as node features. DistilBert [27] is used as the language encoder.

(3) **GIANT**: GIANT [5] is a self-supervised node feature extraction framework that aims to extract more efficient node features for text-attributed graphs. For fair benchmarking, LLM-provided pseudo-labels are used to fine-tune the text encoder of GIANT.

(4) **GLEM**: GLEM [44] is an efficient TAG learning framework by fusing graph structure and language learning with a variational Expectation Maximization (EM) framework. LLM-provided pseudo-labels are used as labels to train GLEM. Following the original work, RevGAT is used as the GNN backbone.

(5) **SIMTEG**: SIMTEG [8] is a TAG learning framework via supervised fine-tuning of the LM encoder of node features. Following the original work, GraphSAGE is used as the GNN backbone.

For all methods, GPT-3.5 Turbo (1106) is used as annotators/teachers.

Training and Implementation Details. We follow the standard inductive learning setting [40], all links between the training nodes and test nodes in the graphs are removed in the training stage to ensure all test nodes are unseen during training. The implementation of the proposed method utilized the PyG and Huggingface libraries, which are licensed under the MIT License. All experiments were conducted on a single NVIDIA RTX 3090 GPU with 24GB VRAM. For pre-training language models, we use a max token number of 512 for full-text features and 48 for keyword features for all three pre-trained language models including bert-base-uncased, distilbert-base-uncased, and microsoft/deberta-base. For the GNN models including GCN, GAT and SAGE, we test on 2-layer models with a hidden dimension of 256. In the training process, we first train the interpreter model alone until convergence, then fix the interpreter model to train the student model. The interpreter model is trained first, then we freeze it and train the student model. We adopt a learning rate of 1e-5 to fine-tune all the pre-trained language models and train for 10 epochs. We train all GNN models for 200 epochs with a learning rate of 0.01.

5.2 Main Results of Label-Free Node Classification

We first compare our proposed method with other test-time LLM-free TAG learning methods. The main results are shown in Table 1. Generally, our proposed framework achieves the best performance on all datasets. More specifically, we have the observations as follows.

Best performance on all datasets. Our proposed method achieves the highest accuracy on all datasets, demonstrating its general efficacy. Compared to the second-best scores, our proposed method yields an over 10.3% performance improvement on
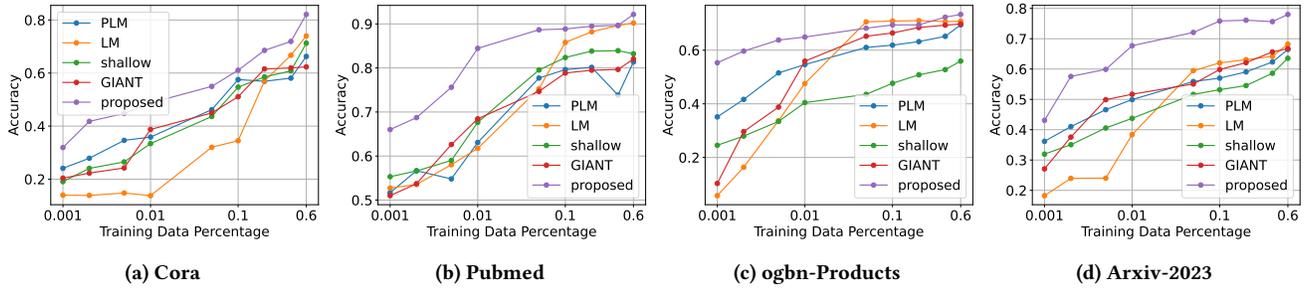
Figure 3: Test accuracy on different proportions of available training data on four datasets. The x-axis represents the percentage of the training data, ranging from 0.001 to 0.6 of the dataset, plotted in the logarithm scale. The y-axis represents the test accuracy.

Table 3: Performance comparison between standard supervised learning and using proposed method as model pre-training on Cora dataset.

Method	GCN	GAT	SAGE
w/o Pre-training	0.8778±0.0084	0.8750±0.0137	0.8815±0.0093
w/ Pre-training	0.8882±0.0020	0.8813±0.0015	0.8910±0.0028

Table 4: Computational cost comparison of different methods. #tokens column denotes the total number of tokens required to pass to and receive from LLMs.

LLM's Role	#tokens	Method	Training time	Test time	Avg. Acc.
Annotator	1.4M	LM	118s	21s	0.7380
		GNN (PLM)	34s	20s	0.6689
		GIANT	470s	21s	0.7194
Teacher	4.2M	Proposed	262s	21s	0.8228

Cora, 2.2% improvement on PubMed, 4% improvement on ogbn-products and 8.3% improvement on arxiv-2023, showing our proposed method can consistently beat the all the comparison methods with significant improvements. Especially, Arxiv [13] is a dataset that ranges out of the training data of GPT-3.5 Turbo, which stresses the concern that the performance increase may due to popular datasets are seen in LLM's training stage.

LM-based methods beat traditional GNN-based methods on the label-free setting. Among our comparison methods, we found pure LM methods achieve performance better than some of the GNN-based methods like GNN (PLM) and GIANT, and yet comparable performance as SIMTEG. This observation is different from the traditional supervised learning setting, where GNN-based methods typically perform better. This discrepancy is possibly due to the accumulation of inaccuracies in LLM's pseudo-labels within GNN-based methods. The results suggest that when using LLM's pseudo labels for training (with a larger noise level), LM-based methods can give a more robust performance. Our method, utilizing the LLM's information as well as the semantic rationales, also benefits from such capability.

Potential as pre-training in standard supervised learning. We also demonstrate that our proposed framework can be used as an effective model pre-training method when the downstream task is a standard supervised learning setting on the same data. We conducted experiments on Cora dataset to validate the performance increase. The results are shown in Table. 3. Results reveal that the proposed distillation can serve as an effective model pre-training in the supervised learning setting. It is worth noting that this pre-training uses the same set of TAG data as the supervised learning setting.

5.3 Efficiency Study

To further analyze the efficiency of the proposed method, we conduct an efficiency study focusing on the performance of our method on different proportions of available training data, and the training and test time.

Training Data Efficiency. We conduct the training data efficiency analysis to explore our method's performance when training data is limitedly available. The results are shown in Fig. 3. Specifically, we adjust the range of the percentage of available training samples in the whole dataset from 0.1% to 60% and plot the test accuracy curves. Results show that our proposed distillation method performs significantly better than all comparison methods when training data is scarce. This reveals distilling the rationale behind predictions can help the model better learn the rule with limited data, validating the effectiveness of our rationale distillation. Results also reveal our method consistently delivers the best performance across almost all training data percentages, although the second-best method may vary. This demonstrates the effectiveness of our approach.

Computational Cost. As a method designed for LLM-free prediction, we further evaluate the computational cost of our method comparing to other method using LLM-provided labels. Results are shown in Table 4. We evaluate the total number of tokens that require to pass to LLM (#tokens), the offline training time and test time of the models, and the test accuracy. Results show that although our model needs to pass more tokens to LLMs to get the rationales compared to only getting pseudo-labels, the training and test time is comparable to other methods. Our method achieves

Table 5: Ablation study on feature enhancements, showing interpreter model accuracy. Performances are distinguished: best (bolded), <u>second</u>, and <u>third</u>. Includes a *Rank.* column for average accuracy ranking across three GNN backbones.

Method	GCN	GAT	SAGE	Rank.
Baseline	0.8090±0.0174	0.8058±0.0320	0.8196±0.0294	6
w/o soft labels	0.8310±0.0376	<u>0.8292±0.0451</u>	**0.8369±0.0527**	3
w/o keywords	<u>0.8363±0.0291</u>	<u>0.8344±0.0288</u>	0.8322±0.0312	2
w/o key edges	0.8247±0.0271	0.8215±0.0302	<u>0.8353±0.0494</u>	5
w/o messages	**0.8379±0.0388**	0.8233±0.0410	0.8296±0.0538	4
Proposed	<u>0.8336±0.0288</u>	**0.8376±0.0209**	<u>0.8326±0.0210</u>	1

Table 6: Ablation study of proposed feature discrepancy feature alignment. The method *Interpreter* denotes the performance of the interpreter model, which is the target (upper bound) for the student model to emulate through alignment.

Method	GCN	GAT	SAGE
Interpreter	0.8336±0.0288	0.8376±0.0209	0.8326±0.0210
Baseline	0.8090±0.0174	0.8058±0.0320	0.8196±0.0294
Proposed-T	0.8127±0.0235	0.8104±0.0157	0.8187±0.0307
Proposed-N	0.8146±0.0196	0.8081±0.0116	0.8204±0.0304
Proposed	**0.8237±0.0187**	**0.8237±0.0137**	**0.8210±0.0296**

significantly better results with the cost of more input and output of LLMs, which is exactly our designation purpose of leveraging more information from LLMs.

5.4 Ablation Study

In the ablation study, we conduct experiments to evaluate 1) each enhanced feature's contribution to the interpreter model, and 2) the improvement of our alignment method compared to the vanilla alignment of soft labels.

Interpreter Model's Feature Enhancements. We conduct an ablation study of each enhanced feature on Cora dataset. The results are shown in Table 5. To construct ablated versions, we remove the LLM-generated soft labels as supervision (w/o soft labels), keywords recognition (w/o keywords), key edges recognition (w/o key edges), and key messages (w/o messages), correspondingly. The results validate the effectiveness of each feature enhancement for the interpreter model. Specifically, the results show that the proposed method with all components achieves the best average performance (ranked first). This reveals using full components can lead to a more reliable performance. The results also reveal that key edges and LLM-generated messages play a relatively larger role in the interpreter model's performance improvement, which is aligned with the nature of TAGs that structure-related information plays a more important role than pure text information.

Ablation Study of Proposed Model Alignment. We further study the effectiveness of two proposed model alignment terms on the Cora dataset. The results are shown in Table 6. In this experiment, the baseline method (denoted by "Baseline") is vanilla model alignment by minimizing the soft predictions, "Proposed-T"
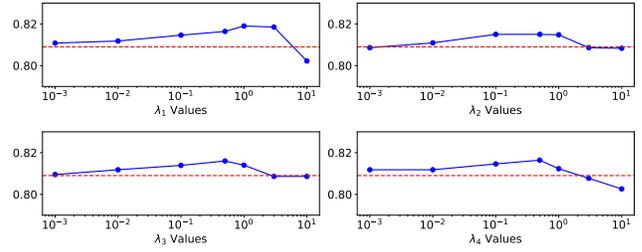

Figure 4: Sensitivity analysis of parameters λ_1, λ_2, λ_3 and λ_4. The red line denotes the baseline performance.

and "Proposed-N" means removing the term of semantics and structure alignment, correspondingly. Results show that the proposed method consistently beat all the ablation versions, thus validating the effectiveness of the two proposed alignment terms. Also, the narrower performance gap between the student model and the interpreter model shows our proposed framework can well infer unseen data with a more consistent performance compared to when LLMs are available.

5.5 Parameter Sensitivity Analysis

We explore the sensitivity of the hyper-parameters in our framework, namely λ_1, λ_2, λ_3, λ_4. For each hyper-parameter, we train the model on seven values ranging from 0.001 to 10, and test the performance of node classification accuracy. Experiments are done on Cora dataset with GCN as the model backbone. The results are shown in Fig. 4. Generally, the results demonstrated the robustness of the proposed framework against the varying of its hyper-parameters. To be more specific, the two parameters corresponding to our proposed alignment terms, λ_2 and λ_3, show a higher stability of the model performance against the change of their values. That shows the stableness of our proposed model alignment method. The parameter λ_1, which corresponds to the loss calculated between the interpreter model-predicted soft labels and LLM-generated soft labels, shows a large performance drop when its value reaches 10. This can be explained as the LLM-generated soft labels are not fully accurate.

6 CONCLUSION

In this paper, we propose a novel framework to distill LLMs to graph models on TAG learning which allows training with both predictions and rationales. We convert the text rationales to multi-level graph rationales and train an interpreter model to bridge LLM's rationales to the graph model and align the interpreter model based on the nature of TAG data. On four datasets, our proposed distillation method overperforms the baseline by an average of 6.2%.

REFERENCES

[1] Guangji Bai, Zheng Chai, Chen Ling, Shiyu Wang, Jiaying Lu, Nan Zhang, Tingwei Shi, Ziyang Yu, Mengdan Zhu, Yifei Zhang, et al. 2024. Beyond efficiency: A

systematic survey of resource-efficient large language models. *arXiv preprint arXiv:2401.00625* (2024).

[2] Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. 2024. LLaGA: Large Language and Graph Assistant. *arXiv preprint arXiv:2402.08170* (2024).

[3] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393* (2023).

[4] Zhikai Chen, Haitao Mao, Hongzhi Wen, Haoyu Han, Wei Jin, Haiyang Zhang, Hui Liu, and Jiliang Tang. 2023. Label-free node classification on graphs with large language models (llms). *arXiv preprint arXiv:2310.04668* (2023).

[5] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. 2021. Node feature extraction by self-supervised multi-scale neighborhood prediction. *arXiv preprint arXiv:2111.00064* (2021).

[6] Sayantan Dasgupta, Trevor Cohn, and Timothy Baldwin. 2023. Cost-effective distillation of large language models. In *Findings of the Association for Computational Linguistics: ACL 2023*. 7346–7354.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. 2023. Simteg: A frustratingly simple approach improves textual graph learning. *arXiv preprint arXiv:2308.02565* (2023).

[9] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[10] Daniel Hajialigol, Hanwen Liu, and Xuan Wang. 2023. XAI-CLASS: Explanation-Enhanced Text Classification with Extremely Weak Supervision. *arXiv preprint arXiv:2311.00189* (2023).

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[12] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654* (2020).

[13] Xiaoxin He, Xavier Bresson, Thomas Laurent, and Bryan Hooi. 2023. Explanations as Features: LLM-Based Features for Text-Attributed Graphs. *arXiv preprint arXiv:2305.19523* (2023).

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[15] Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071* (2022).

[16] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301* (2023).

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[18] Yuntong Hu, Zheng Zhang, and Liang Zhao. 2023. Beyond Text: A Deep Dive into Large Language Models' Ability on Understanding Graph Data. *arXiv preprint arXiv:2310.04944* (2023).

[19] Jin Huang, Xingjian Zhang, Qiaozhu Mei, and Jiaqi Ma. 2023. Can llms effectively leverage graph structural information: when and why. *arXiv preprint arXiv:2309.16595* (2023).

[20] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351* (2019).

[21] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[22] Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023. Symbolic Chain-of-Thought Distillation: Small Models Can Also" Think" Step-by-Step. *arXiv preprint arXiv:2306.14050* (2023).

[23] Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, et al. 2022. Explanations from large language models make small reasoners better. *arXiv preprint arXiv:2210.06726*

(2022).

[24] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. 2015. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643* (2015).

[25] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410* (2022).

[26] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3 (2000), 127–163.

[27] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[28] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[29] Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*. 7059–7073.

[30] Shengyin Sun, Yuxiang Ren, Chen Ma, and Xuecang Zhang. 2023. Large Language Models as Topological Structure Enhancers for Text-Attributed Graphs. *arXiv preprint arXiv:2311.14324* (2023).

[31] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2023. Graphgpt: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023* (2023).

[32] Vladimir Vapnik, Rauf Izmailov, et al. 2015. Learning using privileged information: similarity control and knowledge transfer. *J. Mach. Learn. Res.* 16, 1 (2015), 2023–2049.

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[34] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, et al. 2023. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863* 1 (2023).

[35] Zheng Wang, Jialong Wang, Yuchen Guo, and Zhiguo Gong. 2021. Zero-shot node classification with decomposed graph prototype network. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1769–1779.

[36] Lirong Wu, Haitao Lin, Yufei Huang, and Stan Z Li. 2022. Knowledge distillation improves graph structure augmentation for graph neural networks. *Advances in Neural Information Processing Systems* 35 (2022), 11815–11827.

[37] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Eric Sun, and Yue Zhang. 2023. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *arXiv preprint arXiv:2312.02003* (2023).

[38] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2024. Language is all a graph needs. In *Findings of the Association for Computational Linguistics: EACL 2024*. 1955–1973.

[39] Qin Yue, Jiye Liang, Junbiao Cui, and Liang Bai. 2022. Dual Bidirectional Graph Convolutional Networks for Zero-shot Node Classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2408–2417.

[40] Shichang Zhang, Yozen Liu, Yizhou Sun, and Neil Shah. 2021. Graph-less neural networks: Teaching old mlps new tricks via distillation. *arXiv preprint arXiv:2110.08727* (2021).

[41] Yifei Zhang, Siyi Gu, Bo Pan, Guangji Bai, Xiaofeng Yang, and Liang Zhao. 2023. Visual Attention-Prompted Prediction and Learning. *arXiv preprint arXiv:2310.08420* (2023).

[42] Yifei Zhang, Bo Pan, Chen Ling, Yuntong Hu, and Liang Zhao. 2024. ELAD: Explanation-Guided Large Language Models Active Distillation. *arXiv preprint arXiv:2402.13098* (2024).

[43] Zheng Zhang, Yuntong Hu, Bo Pan, Chen Ling, and Liang Zhao. 2024. TAGA: Text-Attributed Graph Self-Supervised Learning by Synergizing Graph and Text Mutual Transformations. *arXiv preprint arXiv:2405.16800* (2024).

[44] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2022. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709* (2022).

[45] Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. 2021. Textgnn: Improving text encoder via graph neural network in sponsored search. In *Proceedings of the Web Conference 2021*. 2848–2857.