Leveraging Large Language Models for Node Generation in Few-Shot Learning on Text-Attributed Graphs

Jianxiang Yu[1], **Yuxiang Ren**[2*], **Chenghua Gong**[1], **Jiaqi Tan**[1], **Xiang Li**[1,3,4*], **Xuecang Zhang**[2]
[1]East China Normal University
[2]Advance Computing and Storage Lab, Huawei Technologies
[3] Engineering Research Center of Blockchain Data Management (East China Normal University), Ministry of Education
[4] Shanghai Engineering Research Center of Big Data Management
{jianxiangyu, chenghuagong, jiaqitan}@stu.ecnu.edu.cn, {renyuxiang1,zhangxuecang}@huawei.com,
xiangli@dase.ecnu.edu.cn

Abstract

Text-attributed graphs have recently garnered significant attention due to their wide range of applications in web domains. Existing methodologies employ word embedding models for acquiring text representations as node features, which are subsequently fed into Graph Neural Networks (GNNs) for training. Recently, the advent of Large Language Models (LLMs) has introduced their powerful capabilities in information retrieval and text generation, which can greatly enhance the text attributes of graph data. Furthermore, the acquisition and labeling of extensive datasets are both costly and time-consuming endeavors. Consequently, few-shot learning has emerged as a crucial problem in the context of graph learning tasks. In order to tackle this challenge, we propose a lightweight paradigm called LLM4NG, which adopts a plug-and-play approach to establish supervision signals by leveraging LLMs for node generation. Specifically, we utilize LLMs to extract semantic information from the labels and generate samples that belong to these categories as exemplars. Subsequently, we employ an edge predictor to capture the structural information inherent in the raw dataset and integrate the newly generated samples into the original graph. This approach harnesses LLMs for enhancing class-level information and seamlessly introduces labeled nodes and edges without modifying the raw dataset, thereby facilitating the node classification task in few-shot scenarios. Extensive experiments demonstrate the outstanding performance of our proposed paradigm, particularly in low-shot scenarios. For instance, in the 1-shot setting of the ogbn-arxiv dataset, LLM4NG achieves a 76% improvement over the baseline model.

Introduction

Text-Attributed Graphs (TAGs) are prevalent in a variety of real-world scenarios, such as product networks, social networks, and citation networks (Yao, Mao, and Luo 2019; Nguyen et al. 2020). In TAGs, nodes represent entities with textual information and edges capture relationships between entities. For example, amazon product network is one of the web applications and in its review datasets (Ni, Li, and McAuley 2019), each product can be represented as a node, featured with various text attributes, such as descriptions and brands. By constructing edges between nodes based on user purchase history, TAGs effectively capture the relationships among different products. TAGs are a powerful representation of data that combines textual information with graph structures. The utilization of TAGs empowers us to unlock new discoveries across various domains, including recommendation systems (Zhu et al. 2021) and fake news detection (Benamira et al. 2019).

Graph Neural Networks (GNNs) currently utilize graph structure and node features to learn the representation of nodes through message propagation strategies. Many GNNs adopt a naive way to encode the textual information of nodes in TAGs as non-contextualized shallow embeddings (Miaschi and Dell'Orletta 2020) e.g., Bag-of-Words (Harris 1954) and TF-IDF (Salton and Buckley 1988) embeddings. However, the resulting embeddings are unable to capture polysemy and the semantic relationships between words. Recent advancements in natural language processing (NLP) have introduced contextualized word embeddings such as BERT (Devlin et al. 2018) and Deberta (He et al. 2020). In particular, Sentence-BERT (Reimers and Gurevych 2019) performs better in sentence-level text representations. These language models (LMs) capture the contextual information of words and sentences, leading to more powerful representations of text. There have been some GNNs that utilize both LMs and GNNs for training, combining text features and graph topology to obtain effective node representations (Zhao et al. 2022; Yang et al. 2021).

The emergence of Large Language Models (LLMs) like GPT (Brown et al. 2020), Llama (Touvron et al. 2023) and ChatGLM (Du et al. 2021), has made a significant impact due to their powerful generative capabilities. These models typically have a large number of parameters. They can acquire rich linguistic knowledge and semantic representations through training on large-scale corpora. LLMs use prompt engineering to guide their generation and inference capabilities, mining the extensive knowledge they have learned. As a result, they exhibit exceptional performance across various natural language processing tasks, including machine translation, sentiment analysis, and contextual understanding. Following their success, incorporating LLMs into vari-

*Corresponding authors.

ous applications has received wide attention (Li et al. 2023; Fan et al. 2023). In the field of GNNs, Chen et al. (2023) and He et al. (2023) have made significant progress in exploring the potential of LLMs by enhancing node-level text in different ways. However, such methods can lead to much time consumption on large-scale graphs. This is because constantly invoking LLMs for each node poses efficiency challenges. Furthermore, in the few-shot learning scenarios, the rich node features generated by LLMs could bring only marginal performance improvement due to the scarcity of labeled nodes. Consequently, it is necessary to propose a method that ensures low consumption and is suitable for scenarios with extremely limited labeled data.

Intuitively, as the training set becomes larger, it may contain more diverse information, which enables the models to perform better. We expect to leverage LLMs to establish supervision signals and enhance the training dataset in TAGs, rather than performing data augmentation directly on the nodes. In the field of graph learning, different datasets correspond to various domains. For example, for social networks, we need to know the relationship between user interests and user characteristics, while for citation networks in computer science, we need to know expertise in software, operating systems, etc. Therefore, we need to utilize specific domain knowledge to capture the relationship between samples and labels. LLMs can be regarded as comprehensive "encyclopedias" that encompass domain-specific knowledge across various fields. We can exploit LLMs to provide robust support for models in text-attributed graph learning, helping to establish supervision signals and thus enhance the model performance.

Based on the above, we propose a plug-and-play, lightweight paradigm named LLM4NG that leverages **LLM**s **for** **N**ode **G**eneration. Specifically, the paradigm mines label semantics through LLMs, generating diverse samples from different classes to perform class-level augmentation on the raw dataset, thereby obtaining supervision signals to assist GNNs in downstream few-shot node classification tasks. We simply generate labeled samples from LLMs and insert them into the raw dataset to train them together on any GNNs. In practice, when dealing with node classification tasks in few-shot scenarios of a TAG, using LLMs to generate samples presents two main challenges: 1) How to generate diverse and meaningful labeled samples? 2) How to effectively integrate the generated samples with the raw dataset? Firstly, we utilize prompt engineering techniques to guide LLMs in mining semantic information from labels and generate samples of different classes. The generated samples adhere to the textual format of nodes in the raw dataset. In addition, we utilize different prompt statements to explore the diversity of generated samples and analyze how the quantity of generated samples and LLMs' stochasticity affect the quality of the generated samples. Secondly, we train an edge predictor that uses the raw graph structure as supervision signals to establish connections between the generated nodes and nodes in the raw dataset. In this way, we can seamlessly integrate the generated nodes into the topological structure of the raw dataset, extending the diversity of the dataset and the size of the training set. By employing these strategies, we can effectively address the challenges mentioned above. Next, we can use any GNNs architecture, taking the merged node features and topological structure as input, to train and obtain classification results for the samples from the raw dataset. Using LLMs to generate nodes allows for a certain degree of fault tolerance because the focus is placed on the expertise embedded within the generated samples rather than their authenticity. LLM4NG harnesses LLMs to boost model performance. Finally, our contributions are as follows:

- We propose the use of LLMs for generating samples in graph domain to augment the raw dataset and enrich the training set. To the best of our knowledge, this is the first to use LLMs for node generation. We also provide a new solution for training GNNs in zero-shot scenarios, and it demonstrates superior performance.
- Our proposed paradigm LLM4NG is plug-and-play, requiring only class-level invocation of the LLM and lightweight training on the structure, which can be compatible with any GNNs. Furthermore, we integrate the labeled data generated by LLMs into the raw dataset.
- Extensive experiments have shown that LLM4NG exhibits remarkable performance. While keeping the model and dataset unchanged, our paradigm significantly enhances the model's performance. Notably, in the 1-shot setting of the ogbn-arxiv dataset, LLM4NG achieves a 76% improvement over the baseline model.

Related Work

Language Models on TAGs. When dealing with text-attributed graph data, GNNs often adopt the approach of processing text as shallow non-contextual representations. However, language models have a better understanding of contextual relationships and can express textual content more effectively. Therefore, it becomes necessary to combine language models with graph neural networks to accomplish downstream tasks. For example, GIANT (Chien et al. 2021) extensively exploits the potential connections between graph structure and node attributes. It utilizes graph information to aid in feature extraction while fine-tuning language models. Graphformer (Yang et al. 2021) nests graph aggregation within Transformer blocks for text encoding, enabling a global understanding of the semantic meaning of each node. Further, GLEM (Zhao et al. 2022) trains graph neural networks and language models separately within a variational Expectation-Maximization framework, treating them as a process of knowledge distillation that mutually reinforces each other. With the emergence of LLMs, we can leverage more extensive knowledge and information. TAPE (He et al. 2023) utilizes LLMs to predict the ranked classification list of each node and outputs the corresponding explanations. Additionally, KEA (Chen et al. 2023) describes knowledge entities within the original text of each node to complement the textual content of the nodes. Although these methods have achieved some success, they often come with high computational or invocation costs and may not effectively address the challenges posed by few-shot scenarios with insufficient supervision signals.

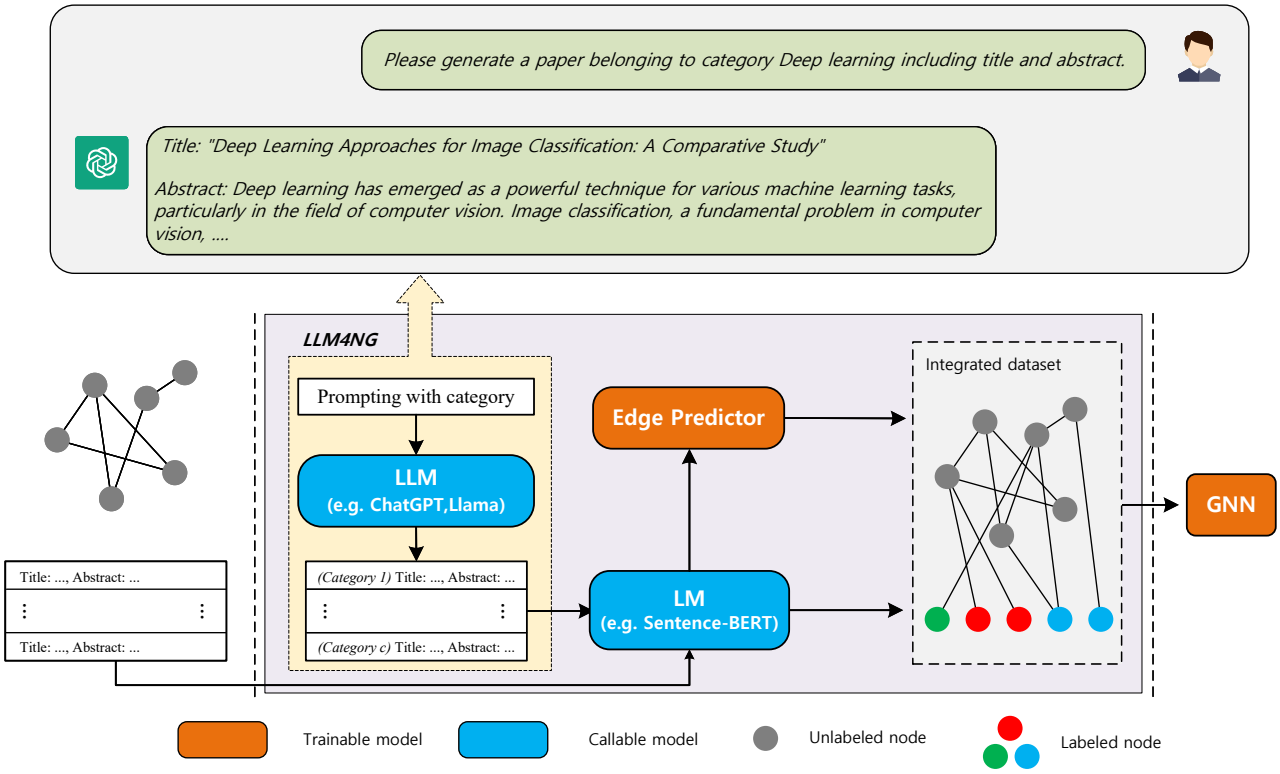

Figure 1: The overall paradigm of LLM4NG.

Few-shot learning on Graphs. In the case of real-world graphs, acquiring labels is a resource-intensive task, making it a formidable challenge in few-shot scenarios. Some methods adopt the paradigm of meta-learning to handle few-shot problems on graphs (Wang et al. 2022b; Lan et al. 2020; Liu et al. 2021; Huang and Zitnik 2020; Liu et al. 2022). In this paper, we focus on the semi-supervised setting, where each category has a few labeled samples. Self-supervised learning methods on graphs learn node representations through a pre-training task and fine-tuning for classification results using a few labeled nodes (Wan et al. 2021a; Zhu et al. 2020; Veličković et al. 2018; Hafidi et al. 2020; Wang et al. 2017). For example, GRACE (Zhu et al. 2020) and GraphCL (Hafidi et al. 2020) construct different views through data augmentation on graphs to bring positive samples closer and push negative samples further apart for learning node representations. Additionally, DGI (Veličković et al. 2018) stands as the pioneering method to introduce contrast between node-level embeddings and graph-level embeddings. This innovation empowers graph encoders to capture both local and global semantic information. MVGRL (Hassani and Khasahmadi 2020) leverages first-order neighbor subgraphs and graph diffusion to generate contrastive views. However, when labeled data is extremely scarce leading to insufficient supervision signals, the model's performance suffers. Several semi-supervised methods have been proposed to enhance model performance using limited labeled information. M3S (Sun, Lin, and Zhu 2020) combines self-training and deep clustering to expand the training set by predicting high-confidence pseudo-labels. CGPN (Wan et al. 2021b) establishes a variational inference framework for Graph Poisson Network (GPN) and GNN models, and integrates contrastive learning to propagate limited label information over the entire graph. Meta-PN (Ding et al. 2022) generates high-quality pseudo-labels through meta-learning strategies to effectively enhance sparse label data. These methods leverage the information of the raw dataset to establish high-confidence pseudo-labels. The advent of LLMs provides us with a novel approach to address few-shot problem. We can harness these LLMs to unearth domain knowledge and generate extra supervision signals.

Preliminary

Text-Attributed Graphs. A Text-Attributed Graph (TAG) is defined as $\mathcal{G}_S = (\mathcal{V}, \mathcal{E}, \mathcal{S}^{\mathcal{V}})$, where $\mathcal{V}$ represents a set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents a set of edges. Each node $v_i \in \mathcal{V}$ is associated with a sequential text $s_i \in \mathcal{S}^{\mathcal{V}}$ and the corresponding label y_i. Each label has a real label text c (e.g., 'Machine Learning' or 'Databases') from the set of all label texts $\mathcal{C}$. The corresponding adjacency matrix of the graph $\mathcal{G}_S$ is denoted by A. For simplicity, we set $A_{ij} = 1$ if $e(v_i, v_j) \in \mathcal{E}$; 0, otherwise.

Few-shot Node Classification. In this paper, we focus on zero- and few-shot node classification, which is one of the most challenging and cutting-edge problems in the field of graph learning. Unlike the few-shot problem in the meta-learning paradigm, in our setting, each class has K labeled samples. Given the labeled $K \times |\mathcal{C}|$ samples, our goal is to predict unlabeled samples in the test set, referred to as K-shot classification. Specifically, zero-shot classification means that there is no labeled sample.

Graph Neural Network Architecture GNNs leverage input graph as the computation graph, aggregating information from a node's neighbors, then updating node representation. Formally, suppose $h_{v_i}^{(l)}$ is the representation of node v_i at the l-th GNN layer, the updating procedure is:

$$h_{v_i}^{(l)} = \underset{\forall v_j \in \mathcal{N}(v_i), \forall e \in \mathcal{E}(v_i, v_j)}{\text{Agg}} \left(\{ \text{Prop}(h_{v_i}^{(l-1)}; h_{v_j}^{(l-1)}, e) \} \right) \quad (1)$$

where $\mathcal{N}(v_i)$ denotes the set of neighbors of node v_i and $\mathcal{E}(v_i, v_j)$ denotes the set of edges connected from node v_j to node v_i. Agg$(\cdot)$ and Prop $(\cdot)$ are the two functions in GNNs. Prop$(\cdot)$ represents the propagation of node information. It uses the target node's representation $h_{v_i}^{(l-1)}$ and the edge e between the two nodes as query, and propagates the information $h_{v_j}^{(l-1)}$ of the source node to the target node v_i. Agg$(\cdot)$ is a differentiable function (e.g., sum, mean, etc.) that aggregates the representations of a node's neighbors.

The Proposed Paradigm

This section introduces details of our proposed paradigm LLM4NG. The general model diagram is shown in Figure 1. We guide LLM through prompt statements to generate some samples with labels and input these generated samples into LM together with the text in the raw dataset to obtain node representations. Subsequently, we perform supervised learning using the raw graph. We utilize the existing edges as supervision signals and input node embeddings to train the edge predictor. After training, we can utilize the embeddings to predict edges between nodes in the raw dataset and generated nodes. The newly generated nodes, along with their structural information, can be integrated into the original dataset. Finally, we can train the entire dataset using any GNNs and obtain results for node classification.

Sample Generation from LLM

When dealing with node classification tasks of a TAG in few-shot scenarios, we lack sufficient supervision signals to make accurate predictions. To address this, we can leverage the textual information within the set of label texts $\mathcal{C}$ to explore the semantics embedded in LLMs and generate M relevant sample instances for each label. We generate a total of $M \times |\mathcal{C}|$ samples, $M \times |\mathcal{C}| \ll |\mathcal{V}|$, which is less costly compared to node-level augmentation. Specifically, we denote c as one of the label texts. Prompt(c) denotes the prompt statement containing the label text c. We feed it into LLM and then get the generated text s_g:

$$s_g = \text{LLM}(\text{Prompt}(c)), \quad c \in \mathcal{C} \quad (2)$$

The text generated by LLM is based on the knowledge learned from the training corpora. It contains both samples that actually exist in the real world and pseudo-samples that the model creates internally by permutating the knowledge. Although those pseudo-samples do not exist in reality, they are still reliable because our goal is to capture domain knowledge tied to the label. For instance, when performing interest classification of users in a social network, for the label "sports", we can generate user profiles of individuals who are interested in basketball or soccer. Although these users do not actually exist, we can still uncover domain-specific information and capture the information that basketball and soccer related to the "sports" category.

Node Representation Initialization with LM

After generating new samples, we treat each of them as a new node g_i on the TAG. Inspired by (Chen et al. 2023), deep sentence embeddings with GNNs have emerged as a powerful baseline compared to non-contextualized shallow embeddings. Moreover, sentence embedding models offer a lightweight approach to get representations without fine-tuning. Therefore, we adopt sentence embedding models to extract information from the text such as Sentence-BERT (Reimers and Gurevych 2019) and e5-large (Wang et al. 2022a).

$$h_{v_i} = \text{LM}(s_{v_i}), \quad h_{g_i} = \text{LM}(s_{g_i}), \quad (3)$$

where s_{v_i} and s_{g_i} denote the text of the raw dataset samples and the newly generated sample, respectively. h_{v_i} and h_{g_i} denote the corresponding representations of these samples. We denote $H_n = H_v \cup H_g$ as the representation of all nodes.

Edge Predictor

Next, we insert new nodes into the raw graph to establish connections between newly generated nodes g_i and the raw nodes v_j. Edges play a role in message passing in the paradigm of GNNs, and new nodes with labels need to propagate supervision signals to other nodes without labels. We aim to ensure that unlabeled nodes receive as much relevant information from similar samples as possible to assist downstream classification tasks.

We consider this problem from two perspectives. First, we perform a coarse screening of potential edges between new nodes and raw nodes. To achieve this, we filter irrelevant edges by the cosine similarity of the representations in the latent space.

$$e(h_{g_i}, h_{v_j}) \in \mathcal{E}_P, \quad \text{if sim}(h_{g_i}, h_{v_j}) = h_{g_i} h_{v_j}^T > \delta \quad (4)$$

where δ as a hyper-parameter represents the similarity threshold. We add edges that have larger similarity weights than δ to the edge prediction set $\mathcal{E}_P$.

However, relying solely on node similarity to determine edge existence is too simple. It is also expected the newly added edges share high similarity with raw edges in the graph. To achieve this goal, we use the edges in the raw graph as supervision signals and construct an edge predictor for the link prediction task. Specifically, we create a binary classification task where the training set is constructed from

the raw graph, and the test set is the edge prediction set $\mathcal{E}_P$. We take raw edges in the graph as positive samples while randomly sampling an equal number of non-existent edges as negative samples:

$$y_e(h_{v_i}, h_{v_j}) = \begin{cases} 1 & \text{if } e(h_{v_i}, h_{v_j}) \in \mathcal{E} \\ 0 & \text{if } e(h_{v_i}, h_{v_j}) \notin \mathcal{E} \end{cases} \quad (5)$$

Then, we concatenate the representations of two-end nodes in each edge and feed them into a multi-layer perceptron (MLP) to obtain the probability of edge existence $\hat{y}_e$.

$$\hat{y}_e(h_{v_i}, h_{v_j}) = \text{MLP}(h_{v_i} || h_{v_j}) \quad (6)$$

Finally, we treat y_e as ground truth and use the Cross-Entropy loss as the objective function. After training the edge predictor, we input the node pairs from the test set into the model to obtain the edge probabilities for the node pairs in the test set. We then select the top-k edges with the highest probabilities and add them to the raw graph. This integrates the newly added nodes with the raw graph.

LLM4NG

Finally, we derive the new adjacency matrix A_n and the node representations H_n, which are then fed into any GNNs along with the set of labeled nodes $\mathcal{Y}_L$ (a total of $(K + M) * |\mathcal{C}|$ samples) to output the classification results Y_n:

$$Y_n = \text{GNN}(A_n, H_n, \mathcal{Y}_L). \quad (7)$$

The proposed paradigm LLM4NG utilizes LLMs to mine semantic information of labels in few-shot scenarios, enabling sample generation and the establishment of supervision signals. Subsequently, we employ a simple model Edge Predictor to obtain additional structural information. Our proposed approach is lightweight, which allows us to facilitate node classification in few-shot scenarios by simply adding labeled nodes and edges without altering the raw dataset.

Experiments

Datasets and Baselines

To evaluate the performance of LLM4NG, we employ three citation network datasets: Cora (McCallum et al. 2000), Pubmed (Sen et al. 2008), and ogbn-arxiv (Hu et al. 2020). Detailed descriptions and statistics on these datasets are provided in Supplement S1.1. We also compare LLM4NG with 9 other SOTA baselines, which can be categorized into four groups:

• **[Classical base models]**: GCN (Kipf and Welling 2016) and GAT (Veličković et al. 2017) are common benchmark models in graph learning for node-level representation.

• **[Graph self-supervised models]**: DGI (Veličković et al. 2018), MVGRL (Hassani and Khasahmadi 2020), and GRACE (Zhu et al. 2020) are prominent benchmark approaches in graph neural networks for self-supervised learning tasks.

• **[Traditional graph data augmentation methods]**: Mixup (Wang et al. 2021) and DropEdge (Rong et al. 2019) are classic methods for augmenting graph data.

• **[Graph models in few-shot scenarios]**: CGPN (Wan et al. 2021b) and Meta-PN (Ding et al. 2022) are proposed for few-shot scenarios.

When running GCN and GAT, we execute the features built into PyG (Fey and Lenssen 2019) and the sentence embeddings obtained from LM each once. Further, for more details on the dataset and baseline, see Appendix A.

Experimental Settings

In LLM4NG, we adopt Sentence-BERT (Reimers and Gurevych 2019) (MiniLM) as LM to obtain sentence embeddings. As for LLM, we use ChatGPT (Brown et al. 2020) (i.e., "gpt-3.5-turbo") and a basic form of prompt in Table S2 of Supplement S2 to generate samples. We set the number of generated samples per class M to be 10 across all datasets. The similarity threshold δ is chosen from 0.1 to 0.8 with step size 0.1. Regarding the selection of top-k node pairs in the set of edge prediction set $\mathcal{E}_P$, we fine-tune the values k to be $1\times$, $5\times$, $10\times$, $30\times$, $50\times$, $70\times$, $100\times$, and $200\times$ of the number of generated samples $M|\mathcal{C}|$. We use the integrated new graph as input data and conduct experiments on different shot scenarios using two backbone models, GCN and GAT, to verify the effectiveness of our proposed paradigm. The notation "w/o A" indicates the absence of an edge predictor module, meaning that the generated labeled samples are treated as isolated points during model training. For fair comparison, we report the average results on the test set with standard deviations of 5 runs for all experiments. We run all the experiments on a server with 32G memory and a single Tesla V100 GPU. We provide our code and data here: https://github.com/jianxiangyu/LLM4NG

Due to the space limitation, we move the experimental setup (Appendix A.3), case study of different prompt statements (Appendix B), hyperparameter analysis (Appendix C), and instances of generating samples using LLMs (Appendix D) to the appendix.

Performance Comparison

We perform zero- and few-shot node classification on TAGs. We construct a K-shot training set, where each class includes K samples, with K values chosen from the set $\{0, 1, 2, 3, 4, 5, 10\}$. For Cora and PubMed, we randomly sample $K \times |\mathcal{C}|$ nodes from all the nodes as the training set. From the remaining nodes, we randomly select 500 nodes as the validation set and 1000 nodes as the test set. For ogbn-arxiv, we follow its original partitioning, but we reduce the size of the training set to the K-shot setting as specified. The results are reported in Table 1, 2 and 3. From the tables, we have the following observations: (1) In all cases, LLM4NG has demonstrated exceptional performance, particularly in low-shot scenarios. For example, on the ogbn-arxiv dataset, the accuracy of LLM4NG achieves 56.83% in the 1-shot setting, while the runner-up method achieves only 31.42%. We have achieved a remarkable improvement of 76% over the baseline model. Furthermore, our proposed paradigm LLM4NG has significant enhancements on both GCN and GAT backbone models. These results indicate that the supervision signals provided by the generated samples are particularly helpful to the model.

Methods	0-shot	1-shot	2-shot	3-shot	4-shot	5-shot	10-shot
GCN(Pyg)	-	60.32(4.12)	64.86(5.83)	68.30(4.81)	71.94(1.85)	75.74(1.81)	78.16(2.57)
GCN(MiniLM)	-	64.26(6.22)	69.28(5.56)	74.02(4.70)	75.64(2.93)	78.30(1.15)	80.24(1.58)
GAT(Pyg)	-	51.76(4.97)	55.90(4.61)	61.16(5.32)	67.74(2.14)	71.26(3.06)	75.38(3.26)
GAT(MiniLM)	-	60.02(6.24)	67.58(4.18)	72.14(5.46)	74.24(2.37)	76.90(1.61)	79.16(1.20)
DGI	-	62.06(7.40)	71.26(6.02)	72.96(2.88)	76.88(3.50)	78.42(2.11)	79.92(1.25)
MVGRL	-	61.94(10.64)	67.48(4.55)	67.68(1.83)	74.66(1.96)	76.98(3.27)	79.16(1.97)
GRACE	-	67.00(9.17)	68.78(3.98)	73.20(5.87)	75.36(2.33)	77.68(2.34)	80.12(1.03)
Mixup+GCN	-	43.56(12.47)	53.58(8.58)	64.38(3.57)	65.58(6.45)	68.72(4.46)	71.50(3.04)
DropEdge+GCN	-	48.67(2.85)	52.73(4.17)	63.03(7.00)	69.33(2.23)	70.13(2.82)	79.55(0.96)
Meta-PN	-	59.20(10.99)	68.92(4.74)	75.30(1.80)	76.50(2.63)	77.10(3.37)	80.12(1.15)
CGPN	-	67.94(4.09)	73.46(2.78)	75.76(0.91)	75.70(1.80)	77.46(2.63)	77.80(0.61)
LLM4NG+GCN	**74.46(1.60)**	**76.48(0.90)**	**76.08(2.75)**	76.26(2.88)	**76.94(2.07)**	79.06(0.75)	80.40(1.01)
LLM4NG+GAT	74.14(1.22)	75.90(1.86)	75.44(2.38)	**76.34(2.63)**	76.40(1.51)	78.14(1.25)	79.60(1.16)
LLM4NG+GCN w/o A	70.44(1.72)	75.12(1.60)	75.24(1.88)	76.04(3.21)	76.86(1.58)	**79.28(1.20)**	**80.94(1.55)**
LLM4NG+GAT w/o A	74.08(1.49)	74.38(2.03)	75.00(4.36)	74.82(3.77)	75.98(3.42)	77.20(3.36)	78.82(2.57)
(improv.)	-	(+12.57%)	(+3.57%)	(+0.77%)	(+0.08%)	(+1.10%)	(+1.02%)

Table 1: Node classification performance in few-shot scenarios of Cora w.r.t. the classification accuracy. Best results are in bold.

Methods	0-shot	1-shot	2-shot	3-shot	4-shot	5-shot	10-shot
GCN(Pyg)	-	60.52(3.15)	63.16(2.45)	63.22(4.59)	65.58(3.58)	68.64(3.15)	75.26(2.48)
GCN(MiniLM)	-	63.02(3.71)	64.74(4.00)	65.82(2.24)	68.40(4.16)	70.06(4.33)	77.72(1.56)
GAT(Pyg)	-	59.72(4.14)	63.48(3.15)	63.76(2.87)	65.88(3.80)	66.92(3.84)	74.64(3.55)
GAT(MiniLM)	-	61.82(4.21)	63.22(5.70)	65.38(2.30)	66.56(2.96)	69.22(1.47)	76.02(1.86)
DGI	-	64.88(7.84)	64.70(9.13)	69.88(3.39)	70.98(3.72)	73.76(3.96)	77.96(0.95)
MVGRL	-	61.78(9.22)	64.7(10.64)	65.42(8.10)	69.36(1.08)	69.58(3.08)	75.16(4.41)
GRACE	-	63.80(1.93)	67.70(6.22)	68.74(6.06)	69.60(5.67)	71.46(7.00)	76.86(3.18)
Mixup+GCN	-	54.14(3.77)	54.48(7.05)	61.22(5.64)	62.30(6.96)	63.08(8.09)	68.08(4.12)
DropEdge+GCN	-	51.77(8.43)	64.76(2.95)	64.35(5.82)	67.43(1.33)	66.43(1.69)	76.70(1.04)
Meta-PN	-	57.52(3.85)	59.56(6.16)	66.60(7.24)	69.52(9.38)	69.66(6.55)	74.28(4.84)
CGPN	-	59.03(10.05)	56.90(9.95)	63.00(11.36)	65.03(4.37)	64.73(7.68)	64.70(3.72)
LLM4NG+GCN	**75.36(2.43)**	**75.06(2.56)**	**76.82(4.30)**	**76.70(3.46)**	**76.74(2.55)**	**78.50(2.56)**	**80.54(0.78)**
LLM4NG+GAT	74.66(1.04)	75.98(2.82)	76.50(1.85)	76.06(2.67)	76.00(3.15)	78.04(2.22)	78.66(1.07)
LLM4NG+GCN w/o A	73.72(2.88)	74.82(2.45)	75.56(3.17)	75.72(3.36)	76.06(2.68)	77.66(2.85)	79.64(1.33)
LLM4NG+GAT w/o A	74.08(1.49)	74.38(2.03)	75.00(4.36)	74.82(3.77)	75.98(3.42)	77.20(3.36)	78.82(2.57)
(improv.)	-	(+15.70%)	(+13.47%)	(+9.76%)	(+8.11%)	(+6.42%)	(+3.31%)

Table 2: Node classification performance in few-shot scenarios of Pubmed w.r.t. the classification accuracy. Best results are in bold.

(2) LLM4NG achieves impressive performance in zero-shot scenarios, surpassing the performance of other methods even in their low-shot scenarios. On PubMed, our model excels across various shot scenarios. Specifically, our model achieves a score of 75.36% in the 0-shot setting, beating the highest score of 73.76% obtained by other models in the 5-shot scenario. This trend is also observed in the ogbn-arxiv dataset. LLM4NG provides a method that can be trained by GNNs in zero-shot scenarios. (3) Compared to 'w/o A' cases, LLM4NG performs better. The structural information between the generated nodes and the nodes in the raw dataset has demonstrated positive effects, which plays a crucial role in improving model performance. This is because the structure facilitates the propagation of supervisory signals, allowing labeled nodes to influence unlabeled ones, thereby enhancing the model's performance.

In conclusion, the strong performance of LLM4NG highlights the effectiveness of leveraging generated samples to enhance the model's learning capabilities, especially when labeled data is extremely scarce.

Case Study

Case Study for Comparison with LLM-based Enhanced Methods In this section, we compare our method with TAPE (He et al. 2023), the existing LLM-based enhanced method. TAPE feeds titles and abstracts of papers into LLMs, obtaining classification predictions and corresponding explanations through prompt statements. The raw text (TA), predictions (P), and explanations (E) are then input into LMs to derive sentence representations. Subsequently, these representations are input into three distinct GNN classifiers, with the final classification result determined by averaging the output probabilities of these classifiers. For fairness, Sentence-Bert (Reimers and Gurevych 2019) is used across all models to directly obtain sentence representations.

Table 4 displays the experimental results on ogbn-arxiv, a large benchmark dataset. LLM4NG demonstrates superior performance in few-shot scenarios. On the other hand, TAPE incurs substantial cost by requiring calls for all samples. In contrast, our method necessitates only 400 calls, achieving

Methods	0-shot	1-shot	2-shot	3-shot	4-shot	5-shot	10-shot
GCN(Pyg)	-	23.38(5.26)	32.98(9.12)	39.77(6.36)	46.23(3.62)	47.95(2.24)	52.27(1.86)
GCN(MiniLM)	-	31.42(3.89)	40.63(10.52)	45.62(7.02)	55.30(4.08)	53.44(2.01)	56.48(1.24)
GAT(Pyg)	-	22.83(4.48)	24.82(6.01)	29.70(3.03)	40.39(4.03)	40.17(3.47)	50.13(2.28)
GAT(MiniLM)	-	32.19(5.17)	41.43(8.28)	45.03(5.66)	53.92(2.35)	51.94(1.80)	54.39(1.22)
DGI	-	25.75(5.97)	32.93(6.30)	36.47(7.78)	40.71(2.42)	42.32(3.11)	45.79(1.27)
MVGRL	-	16.24(1.71)	17.82(2.26)	21.58(4.11)	23.42(1.79)	24.50(1.45)	31.26(1.17)
GRACE	-	29.91(1.73)	35.72(2.35)	38.24(3.62)	38.68(1.11)	37.72(2.11)	43.51(0.91)
Mixup+GCN	-	20.52(5.95)	27.45(5.09)	30.94(5.31)	35.45(4.47)	38.80(3.34)	44.15(2.21)
DropEdge+GCN	-	10.11(7.54)	20.75(3.86)	25.52(4.00)	29.50(4.70)	29.37(9.65)	37.89(2.24)
Meta-PN	-	23.38(6.10)	27.32(2.67)	31.47(0.73)	30.53(6.62)	34.67(3.68)	39.28(4.20)
CGPN	-	OOM	OOM	OOM	OOM	OOM	OOM
LLM4NG+GCN	**54.60(1.27)**	**56.83(1.82)**	**58.32(1.01)**	**58.70(1.54)**	**60.58(0.74)**	**60.83(0.88)**	61.11(1.70)
LLM4NG+GAT	54.35(1.97)	56.29(2.65)	56.76(1.71)	58.04(1.81)	60.03(1.45)	59.95(1.59)	**61.40(1.06)**
LLM4NG+GCN w/o A	53.82(2.02)	55.89(1.30)	57.13(1.14)	58.68(0.41)	59.83(2.03)	58.87(1.40)	60.31(1.74)
LLM4NG+GAT w/o A	54.13(0.71)	55.18(2.28)	56.50(1.75)	57.85(1.77)	59.70(1.72)	57.84(2.56)	59.89(0.91)
(improv.)	-	(+76.55%)	(+40.77%)	(+28.67%)	(+9.55%)	(+13.83%)	(+8.71%)

Table 3: Node classification performance in few-shot scenarios of ogbn-arxiv w.r.t. the classification accuracy. We highlight the best results in bold. OOM denotes the out-of-the-memory error.

	TA	E	TAPE	LLM4NG	TAPE+LLM4NG
0-shot	-	-	-	54.60(1.27)	56.80(0.13)
1-shot	31.42(3.89)	36.16(6.48)	33.21(3.72)	56.83(1.82)	56.93(2.50)
2-shot	40.63(3.52)	41.36(6.48)	41.98(6.76)	58.32(1.01)	58.73(1.37)
3-shot	45.62(7.02)	51.66(7.27)	48.01(9.82)	58.70(1.54)	59.09(1.57)
4-shot	55.30(4.08)	53.10(5.62)	56.62(2.82)	60.58(0.74)	61.24(1.39)
5-shot	53.44(2.01)	54.24(3.34)	57.76(0.70)	60.83(0.88)	61.53(1.29)
Cost	169,343 calls (120.91$)			400 calls (0.25$)	SUM

Table 4: Comparison with LLM-based Enhanced Methods. SUM indicates that the cost is the sum of the two methods.

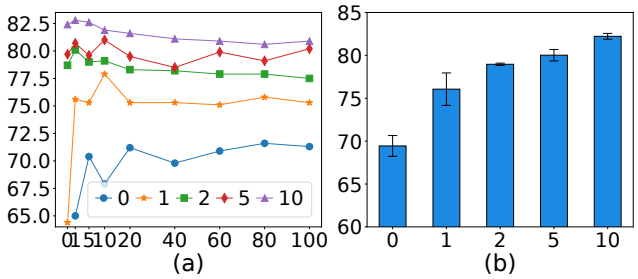

Figure 2: (a) Impact of the quantity of generated samples. The x-axis shows the number of generated samples per label, with different lines for shot numbers. (b) Impact of LLMs' stochasticity on model performance. The x-axis represents shot numbers. In both cases, the y-axis denotes accuracy.

an improvement of over 400 times in terms of both #calls and API cost. It is also worth noting that LLM4NG and TAPE are compatible, and their combination can further improve the model performance.

Case Study of Generated Samples In this section, we analyze the impact of the quantity of generated samples and LLMs' stochasticity on the performance of the model in the Cora dataset. The experiments conducted in this section are run only once and do not involve edge prediction between the generated nodes and the nodes in the raw dataset.

The quantity of generated samples. In this case, we use the same prompt statement LLM4NG-P1 to generate 100×7 samples ($M \times |\mathcal{C}|$), and conduct experiments under different shot settings. Subsequently, we vary the number of samples generated for each class and plot the experimental results in Figure 2(a). We note that as the number of generated samples increases, the accuracy shows an initial increase followed by a decrease, and eventually stabilizes. This phenomenon can be explained by the increased diversity of the generated samples, providing richer supervised signals and improve performance. With an increasing number of generated samples, the additional samples gradually diminish their contribution to the model's performance, and thus the results are not significantly improved.

LLMs' stochasticity. We also repeat the same prompt statement LLM4NG-P1 five times to generate different sample contents, where the number of generated samples M is 10, i.e., a total of 350 ($5 \times 10(M) \times 7(|\mathcal{C}|)$) samples. We ensure that the generated samples are not identical to each other. From Figure 2(b), we observe that although there are variations among the generated samples, the overall quality is quite good, exhibiting strong generalization capabilities. We believe that the effectiveness of the generated samples by LLMs lies in their ability to capture semantic information relevant to the label text. This semantic information is reliable and has a positive impact on classification results.

Conclusion

In this study, we proposed a lightweight paradigm called LLM4NG to address the node classification problem in few-shot scenarios. Specifically, we leveraged the powerful generation capabilities of LLMs to explore the semantic information of labels and generate samples. We then trained an edge predictor to incorporate these generated samples into the raw dataset, thereby providing supervision signals to the model. The impressive experimental results strongly demonstrated the effectiveness of LLM4NG.

Acknowledgments

This work is supported by National Natural Science Foundation of China No. 62202172.

References

Benamira, A.; Devillers, B.; Lesot, E.; Ray, A. K.; Saadi, M.; and Malliaros, F. D. 2019. Semi-supervised learning and graph neural networks for fake news detection. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 568–569.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, Z.; Mao, H.; Li, H.; Jin, W.; Wen, H.; Wei, X.; Wang, S.; Yin, D.; Fan, W.; Liu, H.; et al. 2023. Exploring the potential of large language models (llms) in learning on graphs. *arXiv preprint arXiv:2307.03393*.

Chien, E.; Chang, W.-C.; Hsieh, C.-J.; Yu, H.-F.; Zhang, J.; Milenkovic, O.; and Dhillon, I. S. 2021. Node feature extraction by self-supervised multi-scale neighborhood prediction. *arXiv preprint arXiv:2111.00064*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ding, K.; Wang, J.; Caverlee, J.; and Liu, H. 2022. Meta propagation networks for graph few-shot semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 6524–6531.

Du, Z.; Qian, Y.; Liu, X.; Ding, M.; Qiu, J.; Yang, Z.; and Tang, J. 2021. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*.

Fan, W.; Zhao, Z.; Li, J.; Liu, Y.; Mei, X.; Wang, Y.; Tang, J.; and Li, Q. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046*.

Fey, M.; and Lenssen, J. E. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*.

Hafidi, H.; Ghogho, M.; Ciblat, P.; and Swami, A. 2020. GraphCL: Contrastive Self-Supervised Learning of Graph Representations. arXiv:2007.08025.

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

Hassani, K.; and Khasahmadi, A. H. 2020. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, 4116–4126. PMLR.

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

He, X.; Bresson, X.; Laurent, T.; and Hooi, B. 2023. Explanations as Features: LLM-Based Features for Text-Attributed Graphs. *arXiv preprint arXiv:2305.19523*.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Huang, K.; and Zitnik, M. 2020. Graph meta learning via local subgraphs. *Advances in neural information processing systems*, 33: 5862–5874.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Lan, L.; Wang, P.; Du, X.; Song, K.; Tao, J.; and Guan, X. 2020. Node classification on graphs with few-shot novel labels via meta transformed network embedding. *Advances in Neural Information Processing Systems*, 33: 16520–16531.

Li, J.; Liu, Y.; Fan, W.; Wei, X.-Y.; Liu, H.; Tang, J.; and Li, Q. 2023. Empowering Molecule Discovery for Molecule-Caption Translation with Large Language Models: A ChatGPT Perspective. *arXiv preprint arXiv:2306.06615*.

Liu, Y.; Li, M.; Li, X.; Giunchiglia, F.; Feng, X.; and Guan, R. 2022. Few-shot node classification on attributed networks with graph meta-learning. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 471–481.

Liu, Z.; Fang, Y.; Liu, C.; and Hoi, S. C. 2021. Relative and absolute location embedding for few-shot node classification on graph. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 4267–4275.

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

Miaschi, A.; and Dell'Orletta, F. 2020. Contextual and Non-Contextual Word Embeddings: an in-depth Linguistic Investigation. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, 110–119. Online: Association for Computational Linguistics.

Nguyen, V.-H.; Sugiyama, K.; Nakov, P.; and Kan, M.-Y. 2020. Fang: Leveraging social context for fake news detection using graph representation. In *Proceedings of the 29th ACM international conference on information & knowledge management*, 1165–1174.

Ni, J.; Li, J.; and McAuley, J. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. 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)*, 188–197. Hong Kong, China: Association for Computational Linguistics.

Reimers, N.; and Gurevych, I. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*.

Salton, G.; and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5): 513–523.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Sun, K.; Lin, Z.; and Zhu, Z. 2020. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 5892–5899.

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

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341*.

Wan, S.; Pan, S.; Yang, J.; and Gong, C. 2021a. Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 10049–10057.

Wan, S.; Zhan, Y.; Liu, L.; Yu, B.; Pan, S.; and Gong, C. 2021b. Contrastive graph poisson networks: Semi-supervised learning with extremely limited labels. *Advances in Neural Information Processing Systems*, 34: 6316–6327.

Wang, C.; Pan, S.; Long, G.; Zhu, X.; and Jiang, J. 2017. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 889–898.

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

Wang, S.; Ding, K.; Zhang, C.; Chen, C.; and Li, J. 2022b. Task-adaptive few-shot node classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1910–1919.

Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; and Hooi, B. 2021. Mixup for node and graph classification. In *Proceedings of the Web Conference 2021*, 3663–3674.

Yang, J.; Liu, Z.; Xiao, S.; Li, C.; Lian, D.; Agrawal, S.; Singh, A.; Sun, G.; and Xie, X. 2021. GraphFormers: GNN-nested transformers for representation learning on textual graph. *Advances in Neural Information Processing Systems*, 34: 28798–28810.

Yao, L.; Mao, C.; and Luo, Y. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 7370–7377.

Zhao, J.; Qu, M.; Li, C.; Yan, H.; Liu, Q.; Li, R.; Xie, X.; and Tang, J. 2022. Learning on large-scale text-attributed graphs via variational inference. *arXiv preprint arXiv:2210.14709*.

Zhu, J.; Cui, Y.; Liu, Y.; Sun, H.; Li, X.; Pelger, M.; Yang, T.; Zhang, L.; Zhang, R.; and Zhao, H. 2021. Textgnn: Improving text encoder via graph neural network in sponsored search. In *Proceedings of the Web Conference 2021*, 2848–2857.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2020. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*.