Exploiting Text Semantics for Few and Zero Shot Node Classification on Text-attributed Graph

Yuxiang Wang[1], **Xiao Yan**[1,*], **Shiyu Jin**[1], **Quanqing Xu**[2], **Chuang Hu**[1], **Yuanyuan Zhu**[1], **Bo Du**[1], **Jia Wu**[3] and **Jiawei Jiang**[1,*]
[1] School of Computer Science, Wuhan University
[2] OceanBase
[3] School of Computing, Macquarie University
{nai.yxwang, syjin, handc, yyzhu, dubo, jiawei.jiang}@whu.edu.cn, yanxiaosunny@gmail.com, xuquanqing.xqq@oceanbase.com, jia.wu@mq.edu.au

Abstract

Text-attributed graph (TAG) provides a text description for each graph node, and few- and zero-shot node classification on TAGs have many applications in fields such as academia and social networks. Existing work utilizes various graph-based augmentation techniques to train the node and text embeddings, while text-based augmentations are largely unexplored. In this paper, we propose **T**ext **S**emantics **A**ugmentation (TSA) to improve accuracy by introducing more text semantic supervision signals. Specifically, we design two augmentation techniques, i.e., *positive semantics matching* and *negative semantics contrast*, to provide more reference texts for each graph node or text description. Positive semantic matching retrieves texts with similar embeddings to match with a graph node. Negative semantic contrast adds a negative prompt to construct a text description with the opposite semantics, which is contrasted with the original node and text. We evaluate TSA on 5 datasets and compare with 13 state-of-the-art baselines. The results show that TSA consistently outperforms all baselines, and its accuracy improvements over the best-performing baseline are usually over 5%. The code is at https://github.com/wyx11112/TSA.

1 Introduction

Text-attributed graph (TAG) [Yan *et al.*, 2023] is a prevalent type of graph-structured data, where each node is associated with a text description. For instance, in a citation network, the papers (i.e., nodes) are linked by the citation relations (i.e., edges), and the abstract of each paper serves as the text description. Few-shot and zero-shot node classification on TAGs (FZNC-TAG) predict the categories of the nodes using a few or even no labeled data since labeled data are expensive to obtain [Liu *et al.*, 2021; Liu *et al.*, 2022]. The two tasks have many applications in areas such as recommender system [Gao *et al.*, 2022], social network analysis [Yu *et al.*, 2020], and anomaly detection [Noble and Cook, 2003].

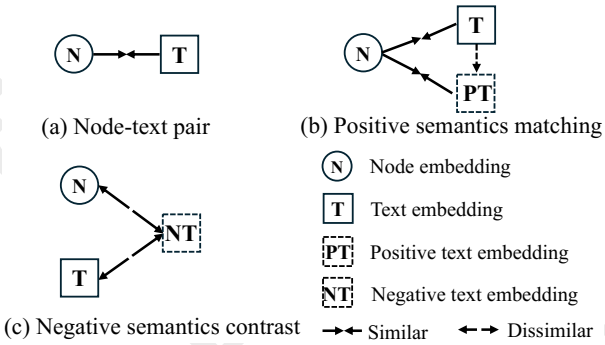

Figure 1: The contrastive loss of G2P2 (a) and two semantic augmentation techniques proposed by TSA (b,c). The node-text pair of G2P2 is specified by data as each node has a text description, and TSA mines more semantic information for nodes and texts.

Existing methods for FZNC-TAG typically follow a two-step process: first learn node and text embeddings on the TAGs and then use prompting to produce classification results [Wen and Fang, 2023; Huang *et al.*, 2023]. They mainly differ in embedding learning and can be classified into three categories. ❶ Self-supervised graph learning methods employ graph augmentation techniques, such as node adding and feature masking, to generate more generalized node embeddings [You *et al.*, 2020; Deng and Hooi, 2021]. They exploit the graph topology but ignores the information in the text semantics. ❷ The two-stage learning methods encode the text using large language models (LLMs) and add the text embedding as additional node features [Tang *et al.*, 2024; Chen *et al.*, 2024]. Then, graph neural network (GNN) methods, e.g., TextGCN [Yao *et al.*, 2019] and GraphSAGE [Hamilton *et al.*, 2017], are used to learn the node embeddings. The limitation of these approaches is that the LLMs are not updated during GNN training [Yan *et al.*, 2023]. ❸ The state-of-the-art end-to-end learning method, G2P2 [Wen and Fang, 2023], jointly trains the GNN and language model via contrastive learning [He *et al.*, 2020]. As shown in Figure 1(a), G2P2 contrasts each node-text pair to ensure that the GNN and language model are aligned in the embedding space.

Existing work utilizes various graph-based augmentation

*Corresponding authors.

to train the node and text embeddings, however, neglecting the semantics information in the texts. As shown in Figure 1(a), G2P2 merely aligns the raw node-text pairs in the TAGs. Furthermore, we observe that the classification accuracy of G2P2 is low. For instance, on the Fitness dataset [Yan *et al.*, 2023], G2P2 only achieves an accuracy of 68.24% and 45.99% for few- and zero-shot classification, respectively. The significantly lower accuracy in zero-shot classification also suggests that the problems are caused by the lack of text semantics and that G2P2's sole reliance on the graph-based augmentation is inadequate for training high-quality models. Thus, we ask the following research question:

> *How to exploit more text semantics to enhance few- and zero-shot classification on TAGs?*

To answer the question, we present TSA, where embedding learning can be improved by enforcing similarity relations among embeddings. Inspired by this idea, we design two augmentation techniques, i.e., positive semantics matching and negative semantics contrast, as shown in Figure 1(b,c). These techniques create additional node-text pairs that have similar or dissimilar embeddings to facilitate model learning.

Positive semantics matching. In Figure 1(b), we provide multiple positive text embeddings for each node embedding. This is achieved by searching the texts that have similar embeddings to the text of the considered graph node. We also encourage the node embedding to be similar to those text embeddings. This augmentation provides more supervision to GNN training and enforces the prior that nodes may belong to the same categories if their texts have similar semantics.

Negative semantics contrast. In Figure 1(c), we pair each text with a semantically opposite *negative text*, which is constructed by adding a learnable negative prompt to the original text. We then encourage the original text embeddings and its corresponding node to be dissimilar to the negative text. This augmentation provides additional text semantics to make the classification robust. For instance, to classify a paper as being related to artificial intelligence, it should not only be similar to the description *"a paper is published at IJCAI"* but also be dissimilar to *"a paper is published at The Lancet"*.

We conduct extensive experiments to evaluate TSA, using 5 datasets and comparing with 13 state-of-the-art baselines. The results show that TSA consistently achieves higher accuracy than all baselines for both few-shot and zero-shot setting. In particular, TSA improves the accuracy and F1 scores of few-shot classification by 4.6% and 6.9% on average, and zero-shot classification by 8.8% and 9.3%, respectively.

In summary, we make the following contributions:

- We observe that prior methods only contrast node-text pairs, thus failing to effectively capture text semantics. To solve this issue, we propose TSA to enhance the semantic understanding in both model pre-training and inference.
- We incorporate two novel augmentation techniques into TSA: positive semantics matching and negative semantics contrast. These techniques create additional node-text pairs by mining more text semantics from diverse perspectives.
- We conduct extensive experiments to evaluate TSA and compare with state-of-the-art baselines, demonstrating that TSA enjoys high model accuracy and training efficiency.

2 Preliminaries

Text-attributed graph. We denote a text-attributed graph (TAG) as $\mathbf{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, in which $\mathcal{V}$, $\mathcal{E}$, and $\mathbf{X}$ are the node set, edge set, and text set, respectively. Take citation network as an example for TAG. Each node $v_i \in \mathcal{V}$ is a paper, interconnected by the edges $e \in \mathcal{E}$ that signify citation relations. Let $\mathbf{x}_i \in \mathbf{X}$ denote the text (i.e., paper abstract) of the i-th node. Each node has a label to indicate the topic of the paper. Since the graph nodes and papers have a strict one-to-one correspondence, node v_i and text $\mathbf{x}_i$ share an identical label.

Few- and zero-shot learning. For few-shot classification, the test dataset encompasses a support set $\mathcal{S}$ and a query set $\mathcal{Q}$. $\mathcal{S}$ comprises C classes of nodes, with K labeled nodes drawn from each class. These nodes can be used to train or fine-tune the classifier, which is then utilized to classify the nodes in $\mathcal{Q}$. Zero-shot node classification is essentially a special case of few-shot classification with $K = 0$. There are no labeled nodes for both training and testing, and classification depends solely on the class names.

Contrastive loss. Recent researches [Wen and Fang, 2023; Zhao *et al.*, 2024] use the contrastive loss to jointly train the graph and text encoders. Specifically, they employ GCN [Kipf and Welling, 2016] as the graph encoder ϕ to encode each node v_i into a node embedding $\mathbf{n}_i$, and adopt Transformer [Vaswani *et al.*, 2017] as the text encoder ψ to map each text $\mathbf{x}_i$ to a text embedding $\mathbf{t}_i$. That is,

$$\mathbf{n}_i = \phi(v_i), \quad \mathbf{t}_i = \psi(\mathbf{x}_i). \tag{1}$$

Then, they use InfoNCE [He *et al.*, 2020] loss $\mathcal{L}_{CL}$ to maximize the similarity between each node $\mathbf{n}_i$ and its corresponding text $\mathbf{t}_i$, while simultaneously minimizing the similarity between node $\mathbf{n}_i$ and other mismatched texts $\mathbf{t}_j$. As shown in part (1) of Figure 2, $\mathcal{L}_{CL}$ is calculated as follows:

$$\mathcal{L}_{CL} = -\frac{1}{|\mathcal{B}|} \sum_{(\mathbf{n}_i, \mathbf{t}_i) \in \mathcal{B}} \log \frac{\exp(\mathrm{sim}(\mathbf{n}_i, \mathbf{t}_i)/\tau)}{\sum_{j \neq i} \exp(\mathrm{sim}(\mathbf{n}_i, \mathbf{t}_j)/\tau)}, \tag{2}$$

where $\mathcal{B}$ is a data batch, $\mathrm{sim}(,)$ is the cosine similarity, and τ is a learnable temperature.

3 The TSA Framework

In this section, we present a novel pre-training and inference framework, named TSA. We start with a overview and follow up with the detailed descriptions of its components.

3.1 Overview

The overall architecture of our framework is illustrated in Figure 2. The model for few-shot consists of a graph encoder and a text encoder, and an extra negative text encoder is included for zero-shot pre-training. We introduce them as follows.

- **Graph encoder** ϕ. We adopt a graph neural network as the encoder to generate the node embedding $\mathbf{n}$.
- **Text encoder** ψ. We choose Transformer [Vaswani *et al.*, 2017] as the text encoder, and it produces a text embedding $\mathbf{t}$ for each text description.

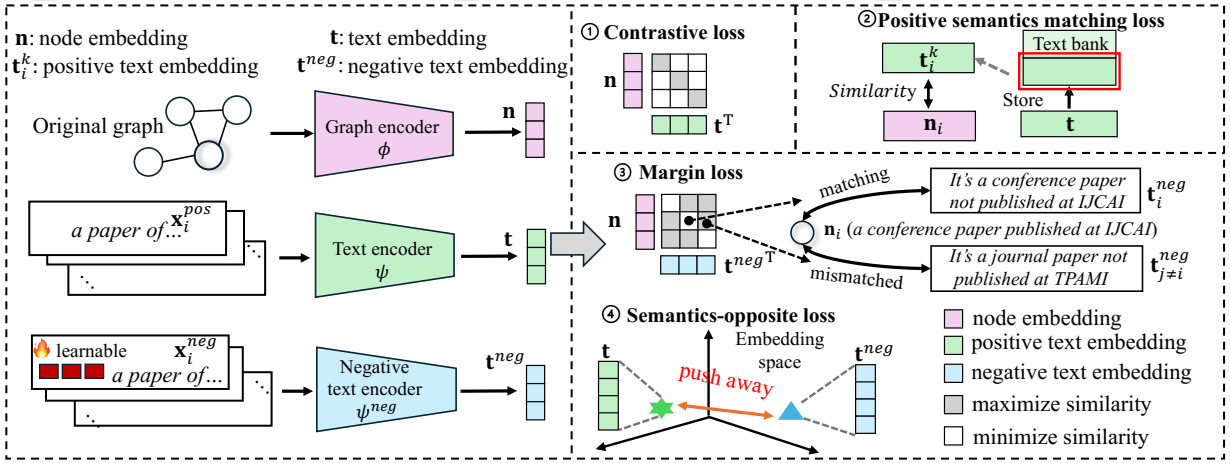

Figure 2: The overview of TSA.

- **Negative text encoder** ψ^{neg}. This maintains the same architecture as the text encoder, with the difference that we train it independently with a negative prompt to generate the negative text presentation $\mathbf{t}^{neg}$.

To effectively train the above encoders, we design two novel loss functions: *positive semantics matching loss* and *negative semantics contrast loss*, which can assist the pre-training model in mining more semantic information. Next, we propose a strategy in Section 3.3, *probability-average*, to enhance zero-shot node classification.

3.2 Text Semantics Augmentation

In this section, we introduce two novel augmentations: negative semantics contrast and positive semantics matching, to exploit text semantics to enhance model training on TAGs.

Positive semantics matching loss. We provide more positive text embeddings for each node embedding. G2P2 [Wen and Fang, 2023] defaults to only one text embedding similar to each node embedding, however there may be multiple similar texts to the target node in TAGs [He *et al.*, 2020; Chuang *et al.*, 2020]. Therefore, we search for multiple text embeddings that are similar to the text embedding of the target node and subsequently encourage the target node embedding to align with these similar text embeddings $\hat{\mathbf{t}}$. The positive semantics matching loss is denoted as follow:

$$\mathcal{L}_{PSM} = -\frac{1}{|\mathcal{B}|}\sum_{(\mathbf{n}_i,\mathbf{t}_i)\in\mathcal{B}} \log\frac{\sum_{k=1}^{\mathbf{K}}\exp(\text{sim}(\mathbf{n}_i,\hat{\mathbf{t}}_i^k)/\tau)}{\sum_{j\neq i}\exp(\text{sim}(\mathbf{n}_i,\mathbf{t}_j)/\tau)}, \quad (3)$$

where $\mathbf{K}$ is the number of similar text embeddings.

However, the above method has two serious drawbacks: first, the complexity of brute-force search for similar text embeddings among all text embeddings is unacceptable; second, storing all text embeddings in GPU memory may lead to out-of-memory. To address these issues, we create a text bank with a capacity of 32K to model the whole text embedding space. As illustrated in the Figure 2(2), whenever a new batch of data arrives, the earliest text embedding is discarded if the capacity of the text bank exceeds a predetermined threshold. Otherwise, it is stored in the bank. Subsequently, we identify the most $\mathbf{K}$ similar text embeddings for target node through similarity calculations. In this way, our text bank is both time-efficient and space-efficient.

Negative prompt. After co-training the graph and text encoder using Equations (2) and (3), the model now possesses the base capability to distinguish node-text pairs. However, understanding the negative semantics within the input text description poses a challenge for the model. For example, we represent a text description such as *"a paper is published at IJCAI"* and its negation *"a paper is published at The Lancet"*. In the embedding space, these two descriptions are likely to be very similar, as their raw texts differ by only few words. To address this issue, we employ negative prompts to generate multiple negative texts that are semantically opposed to the original text descriptions. These negative texts are then used to train a negative text encoder independently. This process helps the negative text encoder learn parameters that are contrary to those of the text encoder.

Our initial idea is to manually construct a series of negative prompts. Specifically, we manually alter the text descriptions by incorporating negation terms such as *"no"*, *"not"*, *"without"*, etc., thus creating a negative prompt corpus that are semantically opposite to the original ones, denoted as $\mathbf{X}^{neg}$. Then, we input the negative text $\mathbf{x}_i^{neg}$ into negative text encoder ψ^{neg} to generate negative text embedding $\mathbf{t}_i^{neg}$.

However, manual modification of the raw text is time-consuming and labor-intensive. To solve this problem, inspired by CoOp [Zhou *et al.*, 2022], we propose a *learnable negative prompt* and add it to the front of raw text. The underlying logic is to represent negative semantics by constantly optimizing the learnable prompt, thereby mirroring the handcrafted negative prompts $\mathbf{X}^{neg}$. Specifically, we concatenate the raw text with M learnable vectors to generate the negative prompt $\mathbf{h}$, allowing $\mathbf{X}^{neg}$ to be replaced by $\mathbf{h}$. Then we input $\mathbf{h}$ into the negative text encoder ψ^{neg}, denoted as follows:

$$\mathbf{h} = [\underbrace{\mathrm{V}_1, \mathrm{V}_2, ...\mathrm{V}_\mathrm{M},}_{\text{negative prompt}}\mathbf{x}], \quad \mathbf{t}_i^{neg} = \psi^{neg}(\mathbf{h}_i), \quad (4)$$

where the negative text encoder is a Transformer with the

same architecture as the text encoder. We further demonstrate from the perspective of information entropy that negative text can replace hand-crafted negative prompt.

Theorem 1. *Given the learnable negative prompt* $\mathbf{h}$ *and hand-crafted negative prompt* $\mathbf{X}^{neg}$*, the information entropy of the learnable negative prompt* $H(\mathbf{h})$ *is related to the lower bound of the information entropy of the hand-crafted negative prompt* $\texttt{LowerBound}(H(\mathbf{X}^{neg}))$ *as follow:*

$$H(\mathbf{h}) \geq \texttt{LowerBound}(H(\mathbf{X}^{neg})). \tag{5}$$

This indicates that learnable negative prompts exhibit higher information entropy than the lower bound of hand-crafted negative prompts. Therefore, learnable negative prompts can effectively capture the negative semantics present in hand-crafted negative prompts. Proof see in Appendix A of [Wang *et al.*, 2025b].

Negative semantics contrast loss. There is still an unsolved problem: *how do we train a negative text encoder?* In other words, how do we ensure that the semantics of the negative text embeddings contradict the original text embeddings. To address this problem, we introduce a novel loss functions, termed negative semantics contrast loss, which comprises margin loss and semantics-opposite loss.

The margin loss anticipates the greatest possible similarity between positive pairs, and conversely, it expects dissimilarity in the case of negative pairs. As shown in Figure 2(3), given a target node v_i, the corresponding negative text description $\mathbf{t}_i^{neg}$ is deemed a negative text, while any other non-corresponding text $\mathbf{t}_{j\neq i}^{neg}$ are considered positive texts. Subsequently, we employ margin loss to assess the degree of matching between the target nodes, positive texts, and negative texts. Specifically, margin loss ensures that the similarity between the target node and the positive text is at least a margin m higher than the similarity with the negative text. Here, we use margin loss instead of InfoNCE loss because margin loss remains constant when the gap between positive and negative samples exceeds m. This is favorable because positive and negative samples are already easily distinguishable. The margin loss $\mathcal{L}_{ML}$ is denoted as follows:

$$\mathcal{L}_{ML} = max(0, 1 + \text{sim}(\mathbf{n}_i, \mathbf{t}_i^{neg}) - \text{sim}(\mathbf{n}_i, \mathbf{t}_{j\neq i}^{neg})) \tag{6}$$

As shown in Figure 2(4), semantics-opposite loss seeks to maximize the mean square error between positive and negative text embeddings. As text $\mathbf{x}_i$ and negative text $\mathbf{t}_i^{neg}$ are semantically opposite, their corresponding embeddings should be as far apart as possible in the text embedding space. We compute the semantics-opposite loss $\mathcal{L}_{SO}$ as follow:

$$\mathcal{L}_{SO} = -\frac{1}{|\mathcal{B}|} \sum_{\mathbf{t}_i \in \mathcal{B}} \|\mathbf{t}_i - \mathbf{t}_i^{neg}\|_2 , \tag{7}$$

where $\|\|_2$ is the L2 norm. Thus, the negative semantics contrast loss is equal to the sum of margin loss and semantics-opposite loss, denoted as $\mathcal{L}_{NSC} = \mathcal{L}_{ML} + \mathcal{L}_{SO}$. It enforces both the node and text embeddings are dissimilar to the corresponding negative text embedding.

Objective. In summary, we denote the total loss of TSA as:

$$\mathcal{L} = \mathcal{L}_{CL} + \mathcal{L}_{PSM} + \alpha\mathcal{L}_{NSC}, \tag{8}$$

where α is the hyperparameter with respect to loss activation. In few-shot pre-training, we do not activate the loss $\mathcal{L}_{NSC}$ (i.e., $\alpha = 0$) because the prompts in few-shot are inherently learnable, incorporating negative prompts would introduce more noise and lead to sub-optimal performance. In contrast, zero-shot classification lacks labeled data during the pre-training. We analyze the ablation experiments on the loss function in detail in Section 4.3.

Complexity Analysis. TSA incorporates both a GNN and a Transformer. The GNN takes $O(LNd^2)$ time for aggregating the neighboring nodes, where L is the network depth, N is the number of nodes and d is the number of dimensions. The Transformer's time complexity is $O(sd^2 + s^2d)$, where s the maximum length of the input sequence. $O(sd^2)$ time is used for mapping vectors at each position to query, key and value vectors, and $O(s^2d)$ time is utilized for the computation of the attention score. Consequently, the overall time complexity of our method is $O(LNd^2 + sd^2 + s^2d)$.

3.3 Prompt Tuning and Inference

Based on the pre-trained model, we tune the model parameters to adapt to the few/zero-shot tasks, which enables classification with a few even no labeled samples while concurrently freezing the pre-trained model's parameters. Prompt tuning is a prompt optimization technique in multimodal models for improving accuracy in few/zero-shot tasks. It uses learnable context vectors $[v]$ to replace the static prompt words. For example, "a photo of a [class]" is replaced by $[v_1][v_2][v_3][class]$. Next, we introduce the foundational paradigm of few- and zero-shot node classification.

Zero-shot classification. In the zero-shot setting, we operate without any labeled samples and rely solely on class name description. To perform C-way node classification, we construct a series of class descriptions $\{\mathbf{D}_c\}_{c=1}^{C}$, via discrete prompts, such as "*a paper of* [*class*]". Then, we input the description text into the pre-trained text encoder to generate the class embedding $\mathbf{g}_c = \psi(\mathbf{D}_c)$. We predict the category of a node v_i by computing the similarity between the node embedding $\mathbf{n}_i$ with the class embedding $\mathbf{g}_c$. The insight behind this is that we align the pre-training and prompting objectives (i.e., to determine whether nodes and texts are similar). Thus, we do not have to tune the parameters of the pre-trained model. The similarity probability between the target node and the candidate class description is calculated as follows:

$$p_i = \frac{\exp(\text{sim}(\mathbf{n}_i, \mathbf{g}_c)/\tau)}{\sum_{c=1}^{C} \exp(\text{sim}(\mathbf{n}_i, \mathbf{g}_c)/\tau)}. \tag{9}$$

Few-shot classification. In the few-shot setting, we conduct a C-way K-shot classification task. Unlike discrete prompts (i.e., "*a paper of ...*") in the zero-shot setting, we have $C \times K$ labeled samples to train learnable prompts. Specifically, we construct a continuous prompt $\mathbf{g}_c$ by adding M learnable vectors to the front of the class description $\mathbf{D}_c$. Formally, we denote $\mathbf{g}_c = \psi([\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_M, \mathbf{D}_c])$. Then, we use Equation (9) to predict the node category, and update the continuous prompts by minimizing the discrepancy between the predicted and ground-truth labels via cross-entropy loss. It is

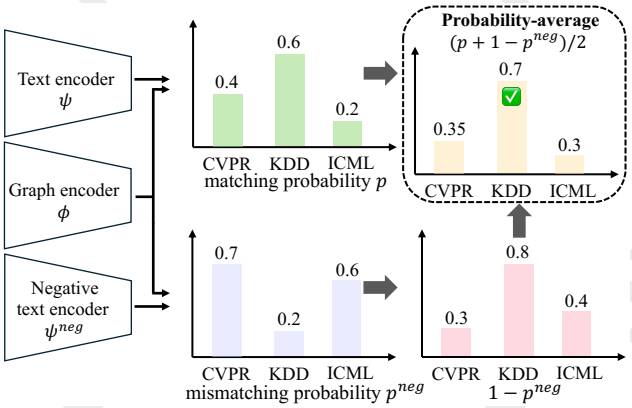

Figure 3: The illustration of probability-average.

worth noting that because $C \times K$ is a small value, the parameters required to fine-tune the prompts are considerably less than those needed for the pre-trained model.

Probability-average. As shown in Figure 3, we propose probability-average to predict node category. Specifically, we first compute p_i by Equation (9). We use the negative text encoder to generate the negative class embedding. Then, we compute negative probability p_i^{neg} by contrasting these negative class embeddings with the target node embedding using Equation (9). p_i denotes the probability that a node belongs to each category and vice versa, p_i^{neg} represents the probability that a node does not belong to each category. Finally, we utilize $(p_i + 1 - p_i^{neg})/2$ to predict the node label. Unlike using a single-encoder model, the negative text encoder provides additional predictive auxiliary for the positive text encoder. Therefore, we use probability-average to balance the output probabilities of the positive and negative text encoders and thus enhancing classification accuracy. Formally, the probability-average strategy can be denoted as follows:

$$\mathcal{Y}_i = \arg\max\ (p_i + 1 - p_i^{neg})/2. \tag{10}$$

Note that the probability-average stragety is only applicable to zero-shot classification, as it requires the negative prompts and negative text encoder to calculate p_i^{neg}. In contrast, few-shot classification directly uses p_i to predict labels.

4 Experimental Evaluation

In this section, we conduct extensive experiments to evaluate TSA and answer the following research questions.

- **RQ1**: How does TSA compare with state-of-the-art methods for few- and zero-shot classification on TAGs?
- **RQ2**: Do our augmentation techniques improve accuracy?
- **RQ3**: How efficient is TSA in training and inference?

4.1 Experiment Settings

Datasets. Following related researches [Yan *et al.*, 2023], we use 5 datasets for experiments. Cora [McCallum *et al.*, 2000] is a citation network, where papers are linked by citation relations and abstract serves as the text. Art, Industrial, M.I., and Fitness are derived from Amazon product categories [Yan *et al.*, 2023], namely, arts, crafts and sewing for Art; industrial and scientific for Industrial; musical instruments for M.I.; and sports-fitness for Fitness, respectively. For the four datasets, an edge is added to construct the graph if a user visits two products successively, and the text is the product description. The five datasets cover different scales (from thousands to millions of nodes) and number of classes (from tens to thousands). These datasets are stored in OceanBase [Yang *et al.*, 2022] and the dataset statistics is provided in Appendix B of [Wang *et al.*, 2025b].

Baselines. We compare TSA with 13 baselines from 5 categories, briefly describe as follow.

- **Supervised GNNs**: GCN [Kipf and Welling, 2016], SAGEsup [Hamilton *et al.*, 2017], TextGCN [Yao *et al.*, 2019]. They are trained in a supervised or semi-supervised manner for the node classification tasks.
- **Self-supervised GNNs**: GraphCL [You *et al.*, 2020], InfoGCL [Xu *et al.*, 2021], PGCL [Lin *et al.*, 2022]. They are first pre-trained via contrastive learning and then fine-tuned for the classification tasks.
- **Graph prompt methods**: GPPT [Sun *et al.*, 2022], GFP [Fang *et al.*, 2024], GraphPromt [Liu *et al.*, 2023]. They reduce the divergence between the pre-training and inference by designing the training objectives and prompts.
- **Language models**: BERT [Devlin *et al.*, 2018], LLM-GNN [Chen *et al.*, 2023], GraphTranslator [Zhang *et al.*, 2024]. Bert is first pre-trained and then fine-tuned for text classification. LLM-GNN and GraphTranslator translate graph into language and predict the labels by LLMs.
- **Co-trained model**: G2P2 [Wen and Fang, 2023]. It employs the contrastive loss to train the GNN and language model jointly such that they produce similar node embedding and text embedding for each node-text pair.

Following G2P2, we use classification accuracy and F1 score to measure performance. We report the average value and standard deviation across 5 runs. Note that we only select language models and G2P2 as the baselines for zero-shot classification, since the other baselines require at least one labeled sample per class for either training or inference.

Environment. During pre-training, we use Adam as the optimizer with a learning rate of 2e-5 for 2 epochs, and the batch size is 64. The number of similar text representations and the capacity of the text bank are set to 1 and 32K, respectively. The length of the learnable negative prompt is 16. Margin m is set to 1. For few-shot pre-training, we do not activate the negative semantics contrast loss, so α is set to 0. Instead, α is set to 0.5 during zero-shot pre-training. Our experiments are conducted on a server with Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz and 8 NVIDIA RTX A6000 48G GPUs.

Task configurations. For few-shot classification, we use a 5-way 5-shot setup, i.e., 5 classes are taken from all classes, and then 5 nodes are sampled from these classes to construct the training set. The validation set is generated in the same way as the training set, and all remaining data is used as the test set. For zero-shot classification, we use 5-way classification, which samples classes but does not provide labeled nodes.

Method	Cora		Fitness		M.I.		Industrial		Art	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
GCN	41.15±2.41	34.50±2.23	21.64±1.34	12.31±1.18	22.54±0.82	16.26±0.72	21.08±0.45	15.23±0.29	22.47±1.78	15.45±1.14
SAGEsup	41.42±2.90	35.14±2.14	23.92±0.55	13.66±0.94	22.14±0.80	16.69±0.62	20.74±0.91	15.31±0.37	22.60±0.56	16.01±0.28
TextGCN	59.78±1.88	55.85±1.50	41.49±0.63	35.09±0.67	46.26±0.91	38.75±0.78	53.60±0.70	45.97±0.49	43.47±1.02	32.20±1.30
GraphCL	76.23±1.80	72.23±1.17	48.40±0.65	41.86±0.89	67.97±2.49	59.89±2.51	62.13±0.65	54.47±0.67	65.15±1.37	52.79±0.83
InfoGCL	78.53±1.12	74.58±1.24	47.56±0.59	41.98±0.77	68.06±0.73	60.64±0.61	52.29±0.66	45.26±0.51	65.41±0.86	53.57±0.75
PGCL	76.32±1.25	73.47±1.53	48.90±0.80	41.31±0.71	76.70±0.48	70.87±0.59	71.87±0.61	65.09±0.47	76.13±0.94	65.25±0.31
GPPT	75.25±1.66	71.16.±1.13	50.68±0.95	44.13±1.36	71.21±0.78	54.73±0.62	75.05±0.36	69.59±0.88	75.85±1.21	65.12±0.83
GFP	75.33±1.17	70.78±1.62	48.61±1.03	42.13±1.53	70.26±0.75	54.67±0.64	74.76±0.37	68.55±0.29	73.60±0.83	63.05±1.61
GraphPrompt	76.61±1.89	72.49±1.81	54.04±1.10	47.40±1.97	71.77±0.83	55.12±1.03	75.92±0.55	70.21±0.28	76.74±0.82	66.01±0.93
BERT	37.86±5.31	32.78±5.01	43.26±1.25	34.97±1.58	50.14±0.68	42.96±1.02	54.00±0.20	47.57±0.50	46.39±1.05	37.07±0.68
LLM-GNN	76.15±0.34	72.31±1.03	63.86±1.85	57.16±1.61	82.19±0.60	75.86±0.30	80.78±0.78	75.12±0.97	78.84±1.29	67.15±1.29
GraphTranslator	79.13±1.38	74.26±0.93	67.55±1.53	56.53±1.71	80.72±0.80	74.20±0.36	82.02±0.61	75.16±0.48	81.01±0.21	69.27±1.27
G2P2	80.08±1.33	75.91±1.39	68.24±0.53	58.35±0.35	82.74±1.98	76.10±1.59	82.40±0.90	76.32±1.04	81.13±1.06	69.48±0.15
TSA	**82.66±0.77**	**79.05±1.25**	**70.79±1.09**	**62.72±1.21**	**87.99±0.64**	**82.61±0.81**	**85.75±0.31**	**80.45±0.25**	**85.55±0.58**	**75.59±0.16**
Gain	+3.2%	+4.1%	+3.7%	+7.5%	+6.3%	+8.6%	+4.3%	+5.4%	+5.4%	+8.8%

Table 1: Accuracy for few-shot node classification (mean±std). The best and runner-up are marked with bold and underlined, respectively. *Gain* is the relative improvement of TSA over the best-performing baseline.

Method	Cora		Fitness		M.I.		Industrial		Art	
	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
BERT	23.56±1.48	17.92±0.86	32.63±1.24	26.58±1.21	37.42±0.67	30.73±0.93	36.88±0.56	29.46±1.12	35.72±1.59	24.10±1.06
LLM-GNN	62.67±1.43	55.21±1.47	42.47±1.01	35.13±1.55	70.81±0.83	62.66±0.84	74.51±1.09	63.54±1.56	74.63±1.46	62.68±1.11
GraphTranslator	62.65±1.85	56.92±0.41	44.07±1.57	38.09±1.89	72.67±1.43	64.72±0.34	74.58±1.31	65.13±1.08	73.77±0.98	61.20±0.65
G2P2	64.35±2.78	58.42±1.59	45.99±0.69	40.06±1.35	74.77±1.98	67.10±1.59	75.66±1.42	68.27±1.31	75.84±1.57	63.59±1.62
TSA	**69.21±1.35**	**61.41±1.82**	**54.41±1.10**	**47.45±1.63**	**79.85±1.35**	**72.58±0.79**	**81.99±0.58**	**73.84±0.33**	**78.22±1.70**	**67.71±0.02**
Gain	+7.6%	+5.1%	+18.3%	+18.4%	+6.8%	+8.2%	+8.4%	+8.2%	+3.1%	+6.5%

Table 2: Accuracy for zero-shot node classification (mean±std). The best and runner-up are marked with bold and underlined, respectively. *Gain* is the relative improvement of TSA over the best-performing baseline.

4.2 Main Results (RQ1)

Few-shot node classification. Table 1 reports the accuracy of TSA and the baselines for few-shot node classification. We can see that TSA consistently outperforms all baselines across the datasets, with an average improvement of 4.6% and 6.9% for classification accuracy and F1 score, respectively. Moreover, the improvements of TSA over the baselines are over than 5% in 6 out of the 10 cases. The results show that TSA can mine text semantics effectively to enhance model pre-training and thus improve accuracy. More explanations are in Appendix C.1 of [Wang *et al.*, 2025b].

Zero-shot node classification. Table 2 reports the accuracy of TSA and the baselines for zero-shot node classification. We only include the language models and G2P2 because the other methods require at least one labeled sample for inference. The results show that TSA consistently outperform all baselines by a large margin. Compared with the best-performing baseline G2P2, the average improvements of TSA in classification accuracy and F1 score are 8.8% and 9.3%, respectively. All methods have lower accuracy for zero-shot classification than few-shot classification because zero-shot classification does not provided labeled samples, and thus the task is more challenging. However, the improvements of TSA are larger for zero-shot classification because it introduces more text semantics for learning.

Robustness to task configuration. We conduct the fewer-way and fewer-shots classification on M.I. dataset. The experimental results are provided in Appendix C.2 of [Wang *et al.*, 2025b]. The results show that TSA outperforms G2P2 across different configurations of ways and shots. We observe that to achieve the same accuracy, TSA requires fewer labeled samples (i.e., shots) than G2P2. Moreover, TSA surpasses the G2P2 across the different ways for zero-shot classification.

Setting	Loss	M.I.	Industrial	Art
Few-shot	$\mathcal{L}_{CL}$	82.74±1.98	82.40±0.90	81.13±1.06
	$\mathcal{L}_{CL+PSM}$	**87.91±0.59**	**85.75±0.31**	**85.37±0.60**
	$\mathcal{L}_{CL+PSM+NSC}$	87.80±0.28	85.63±0.41	85.29±0.66
Zero-shot	$\mathcal{L}_{CL}$	74.77±1.98	75.66±1.42	75.84±1.57
	$\mathcal{L}_{CL+PSM}$	78.32±1.22	81.85±0.55	79.48±1.88
	$\mathcal{L}_{CL+PSM+NSC}$	**79.15±1.35**	**81.99±0.58**	**80.22±1.70**

Table 3: Ablation study of our augmentation techniques. $\mathcal{L}_{CL}$ is the contrastive loss for baseline, *PSM* for positive semantics matching, and *NSC* for negative semantics contrast. Best accuracy in **bold**.

4.3 Micro Experiments

Effect of the augmentations (RQ2). Table 3 presents an ablation study by gradually enabling different augmentations techniques in TSA. We can see that all augmentations are effective in improving accuracy, as adding each of them outperforms the baseline. The best-performing combination for few-shot classification deactivates negative semantics contrast (i.e., $\mathcal{L}_{NSC}$) while zero-shot classification actives $\mathcal{L}_{NSC}$. This is because few-shot classification uses labeled samples to learn the prompt, and the negative prompt learned by $\mathcal{L}_{NSC}$ may interfere with prompt tuning. In contrast, zero-

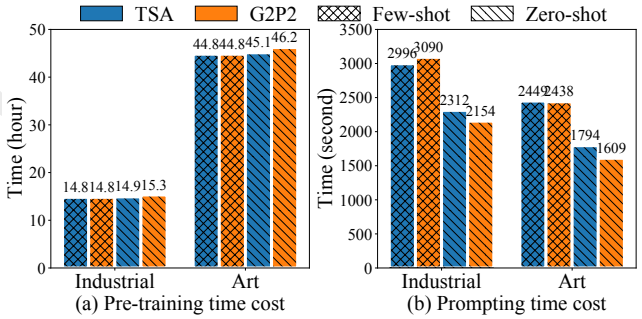

Figure 4: The time cost comparison of pre-training and prompting for G2P2 and TSA.

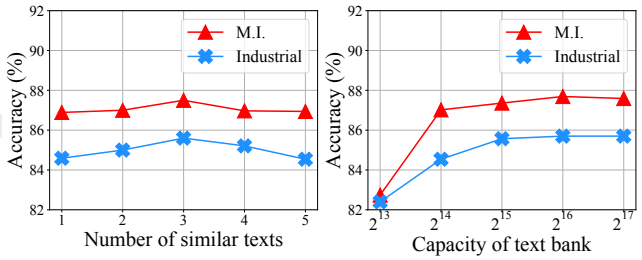

Figure 5: The comparison of the number of similar texts and the capacity of text bank for TSA on M.I. anf Industrial.

shot classification lacks labeled data for prompt tuning, and the $\mathcal{L}_{NSC}$ can provide more text semantics.

Efficiency (RQ3). To examine the efficiency of TSA, we compare with G2P2 for pre-training time and prompting time at inference time. We experiment on Industrial and Art, the two largest datasets, as the running time is shorter on the smaller datasets. Figure 4 shows that TSA and G2P2 have similar pre-training time and prompting time. This is because they both jointly train the GNN and language model, and computing the loss terms has a small cost compared with computing the two models. Few-shot classification has longer prompting time than zero-shot classification because it needs to tune the prompt using the labeled samples.

Hyperparameter sensitivity. We conduct the hyperparameter experiments with respect to the number of similar texts and the capacity of text bank. Figure 5 examines the effect of the two parameters on the Industrial and M.I. datasets. We observe that accuracy first increases but then decreases with the number of similar texts. This is because while more similar texts can provide more text semantics, an excessive number of these signals may introduce noise by including texts that are not truly similar to the target node. Hence, the optimal accuracy are obtained at an intermediate value to balance between semantic supervisions and noises. When increasing the capacity of the text bank, accuracy first increases but then stabilizes. This is because using a larger text bank allows a node to identify texts that are more similar but the similarity will become sufficiently highly when the bank is large enough.

5 Related Work

Graph Pre-training and Prompting. GNNs [Kipf and Welling, 2016; Veličković *et al.*, 2017] use message passing to aggregate features from neighboring nodes to compute graph node embedding. However, early GNN models, such as GCN [Kipf and Welling, 2016], and GAT [Veličković *et al.*, 2017], are supervised and require many labeled nodes for training. To mine supervision signals from unlabeled data, graph self-supervised learning is proposed to train using well-designed pretext tasks. For instance, DGI [Veličković *et al.*, 2018] learns node embeddings by maximizing mutual information between the global and local node embeddings. GPT-GNN [Hu *et al.*, 2020] utilizes a self-supervised graph generation task to combine the graph structural and semantic information.

Graph self-supervised learning [Wang *et al.*, 2024a; Wang *et al.*, 2024b; Liang *et al.*, 2025; Wang *et al.*, 2025a; Zhang *et al.*, 2025] methods still require many labeled to fine-tune specific tasks (e.g., node classification). To further reduce the reliance on labeled instances, graph prompt learning is proposed for few-shot node classification. For example, GPPT predicts the node label by deciding whether an edge exists between the target node and candidate labels. GraphPrompt [Liu *et al.*, 2023] learns embeddings for subgraphs rather than nodes to unify graph-level and node-level tasks. These approaches consider only the graph and thus have limited accuracy for TAGs with text descriptions. To account for the text, TextGCN [Yao *et al.*, 2019] generates text embeddings using pre-trained language models and adds these embeddings as node features for GNN training. G2P2 [Wen and Fang, 2023] jointly trains the language model and GNN with the contrastive strategy and uses prompting for few-shot and zero-shot node classification. However, TSA targets TAGs and considers the graph and text modalities jointly by mining more text semantics while graph pre-training methods consider only the graph.

Pre-trained Language Models (PLMs). PLMs [Devlin *et al.*, 2018; Yang *et al.*, 2022; Yang *et al.*, 2023; Han *et al.*, 2024] enhance the ability to understand natural language by pre-training on large-scale text corpus. The well-known BERT [Devlin *et al.*, 2018], for instance, is pre-trained with two tasks, i.e., masked token reconstruction and next token prediction, to capture contextual information. RoBERTa [Liu *et al.*, 2019] improves BERT by eliminating the next token prediction task, increasing the batch size and data volume during pre-training, and using a dynamic masking strategy. While PLMs achieve great success for text oriented tasks, they cannot capture the topology information for TAGs.

6 Conclusion

In this paper, we study few-shot and zero-shot node classification on text-attributed graphs. We observe that the prior methods is limited to graph-based augmentation techniques, thus we propose TSA as a novel pre-training and inference framework. TSA incorporates two key augmentation techniques, i.e., positive semantics matching and negative semantics contrast, to exploit more text semantics. Extensive experiments show that TSA outperforms existing methods by a large margin. We believe our methodology, i.e., generating node-text pairs that have similar/dissimilar embeddings, is general and can be extended beyond our augmentation techniques.

Acknowledgments

This work was sponsored by National Natural Science Foundation of China (62472327) and the Fundamental Research Funds for the Central Universities (2042025kf0040). This work was supported by Ant Group through CCF-Ant Research Fund (CCF-AFSG RF20240104) and Sichuan Clinical Research Center for Imaging Medicine (YXYX2402).

References

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

[Chen *et al.*, 2024] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61, 2024.

[Chuang *et al.*, 2020] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. *Advances in Neural Information Processing Systems*, 33:8765–8775, 2020.

[Deng and Hooi, 2021] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4027–4035, 2021.

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

[Fang *et al.*, 2024] Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. Universal prompt tuning for graph neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

[Gao *et al.*, 2022] Chen Gao, Xiang Wang, Xiangnan He, and Yong Li. Graph neural networks for recommender system. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, pages 1623–1625, 2022.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

[Han *et al.*, 2024] Fusheng Han, Hao Liu, Bin Chen, Debin Jia, Jianfeng Zhou, Xuwang Teng, Chuanhui Yang, Huafeng Xi, Wei Tian, Shuning Tao, Sen Wang, Quanqing Xu, and Zhenkun Yang. Palf: Replicated write-ahead logging for distributed databases. *Proc. VLDB Endow.*, 17(12):3745–3758, August 2024.

[He *et al.*, 2020] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[Hu *et al.*, 2020] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1857–1867, 2020.

[Huang *et al.*, 2023] Xuanwen Huang, Kaiqiao Han, Dezheng Bao, Quanjin Tao, Zhisheng Zhang, Yang Yang, and Qi Zhu. Prompt-based node feature extractor for few-shot learning on text-attributed graphs. *arXiv preprint arXiv:2309.02848*, 2023.

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

[Liang *et al.*, 2025] Yuxuan Liang, Wentao Zhang, Zeang Sheng, Ling Yang, Quanqing Xu, Jiawei Jiang, Yunhai Tong, and Bin Cui. Towards scalable and deep graph neural networks via noise masking. In *Sponsored by the Association for the Advancement of Artificial Intelligence*, pages 18693–18701, 2025.

[Lin *et al.*, 2022] Shuai Lin, Chen Liu, Pan Zhou, Zi-Yuan Hu, Shuojia Wang, Ruihui Zhao, Yefeng Zheng, Liang Lin, Eric Xing, and Xiaodan Liang. Prototypical graph contrastive learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(2):2747–2758, 2022.

[Liu *et al.*, 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: a robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[Liu *et al.*, 2021] Zemin Liu, Yuan Fang, Chenghao Liu, and Steven CH Hoi. Relative and absolute location embedding for few-shot node classification on graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4267–4275, 2021.

[Liu *et al.*, 2022] Yonghao Liu, Mengyu Li, Ximing Li, Fausto Giunchiglia, Xiaoyue Feng, and Renchu Guan. 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*, pages 471–481, 2022.

[Liu *et al.*, 2023] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, pages 417–428, 2023.

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

[Noble and Cook, 2003] Caleb C Noble and Diane J Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 631–636, 2003.

[Sun *et al.*, 2022] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1717–1727, 2022.

[Tang *et al.*, 2024] Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500, 2024.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

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

[Veličković *et al.*, 2018] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[Wang *et al.*, 2024a] Yuxiang Wang, Xiao Yan, Chuang Hu, Quanqing Xu, Chuanhui Yang, Fangcheng Fu, Wentao Zhang, Hao Wang, Bo Du, and Jiawei Jiang. Generative and contrastive paradigms are complementary for graph self-supervised learning. In *2024 IEEE 40th International Conference on Data Engineering*, pages 3364–3378, 2024.

[Wang *et al.*, 2024b] Yuxiang Wang, Xiao Yan, Shiyu Jin, Hao Huang, Quanqing Xu, Qingchen Zhang, Bo Du, and Jiawei Jiang. Self-supervised learning for graph dataset condensation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 3289–3298, 2024.

[Wang *et al.*, 2025a] Xin Wang, Jiawei Jiang, Xiao Yan, and Qiang Huang. TESA: A trajectory and semantic-aware dynamic heterogeneous graph neural network. In *Proceedings of the ACM on Web Conference*, pages 1305–1315. ACM, 2025.

[Wang *et al.*, 2025b] Yuxiang Wang, Xiao Yan, Shiyu Jin, Quanqing Xu, Chuang Hu, Yuanyuan Zhu, Bo Du, Jia Wu, and Jiawei Jiang. Exploiting text semantics for few and zero shot node classification on text-attributed graph. *arXiv preprint arXiv:2505.08168*, 2025.

[Wen and Fang, 2023] Zhihao Wen and Yuan Fang. Augmenting low-resource text classification with graph-grounded pre-training and prompting. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023.

[Xu *et al.*, 2021] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-aware graph contrastive learning. *Advances in Neural Information Processing Systems*, 34:30414–30425, 2021.

[Yan *et al.*, 2023] Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, et al. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264, 2023.

[Yang *et al.*, 2022] Zhenkun Yang, Chuanhui Yang, Fusheng Han, Mingqiang Zhuang, Bing Yang, Zhifeng Yang, Xiaojun Cheng, Yuzhong Zhao, Wenhui Shi, Huafeng Xi, Huang Yu, Bin Liu, Yi Pan, Boxue Yin, Junquan Chen, and Quanqing Xu. Oceanbase: a 707 million tpmc distributed relational database system. *Proc. VLDB Endow.*, 15(12):3385–3397, August 2022.

[Yang *et al.*, 2023] Zhifeng Yang, Quanqing Xu, Shanyan Gao, Chuanhui Yang, Guoping Wang, Yuzhong Zhao, Fanyu Kong, Hao Liu, Wanhong Wang, and Jinliang Xiao. Oceanbase paetica: A hybrid shared-nothing/shared-everything database for supporting single machine and distributed cluster. *Proc. VLDB Endow.*, 16(12):3728–3740, August 2023.

[Yao *et al.*, 2019] Liang Yao, Chengsheng Mao, and Yuan Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377, 2019.

[You *et al.*, 2020] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.

[Yu *et al.*, 2020] Junliang Yu, Hongzhi Yin, Jundong Li, Min Gao, Zi Huang, and Lizhen Cui. Enhancing social recommendation with adversarial graph convolutional networks. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3727–3739, 2020.

[Zhang *et al.*, 2024] Mengmei Zhang, Mingwei Sun, Peng Wang, Shen Fan, Yanhu Mo, Xiaoxiao Xu, Hong Liu, Cheng Yang, and Chuan Shi. Graphtranslator: Aligning graph model to large language model for open-ended tasks. In *Proceedings of the ACM on Web Conference 2024*, pages 1003–1014, 2024.

[Zhang *et al.*, 2025] Yongliang Zhang, Yuanyuan Zhu, Hao Zhang, Congli Gao, Yuyang Wang, Guojing Li, Tianyang Xu, Ming Zhong, Jiawei Jiang, Tieyun Qian, Chenyi Zhang, and Jeffrey Xu Yu. Tgraph: A tensor-centric graph processing framework. *Proc. ACM Manag. Data*, 3(1):81:1–81:27, 2025.

[Zhao *et al.*, 2024] Huanjing Zhao, Beining Yang, Yukuo Cen, Junyu Ren, Chenhui Zhang, Yuxiao Dong, Evgeny Kharlamov, Shu Zhao, and Jie Tang. Pre-training and prompting for few-shot node classification on text-attributed graphs. *arXiv preprint arXiv:2407.15431*, 2024.

[Zhou *et al.*, 2022] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16825, 2022.