Text-Attributed Graph Learning with Coupled Augmentations

Chuang Zhou[1,*], **Jiahe Du**[1,*], **Huachi Zhou**[1],
Hao Chen[1], **Feiran Huang**[2], **Xiao Huang**[1†]
[1]The Hong Kong Polytechnic University, China; [2]Jinan university, China
{chuang-qqzj.zhou, jiahe.du, huachi666.zhou}@connect.polyu.hk;
sundaychenhao@gmail.com; huangfr@jnu.edu.cn; xiaohuang@comp.polyu.edu.hk

Abstract

Modeling text-attributed graphs is a well-known problem due to the difficulty of capturing both the text attribute and the graph structure effectively. Existing models often focus on either the text attribute or the graph structure, potentially neglecting the other aspect. This is primarily because both text learning and graph learning models require significant computational resources, making it impractical to directly connect these models in a series. However, there are situations where text-learning models correctly classify text-attributed nodes, while graph-learning models may classify them incorrectly, and vice versa. To fully leverage the potential of text-attributed graphs, we propose a Coupled Text-attributed Graph Learning (CTGL) framework that combines the strengths of both text-learning and graph-learning models in parallel and avoids the computational cost of serially connecting the two aspect models. Specifically, CTGL introduces coupled text-graph augmentation to enable coupled contrastive learning and facilitate the exchange of valuable information between text learning and graph learning. Experimental results on diverse datasets demonstrate the superior performance of our model compared to state-of-the-art text-learning and graph-learning baselines.

1 Introduction

Text-attributed graphs are increasingly prevalent in modern society across various domains, including academic networks (Tang et al., 2008), e-commerce platforms (He and McAuley, 2016), and social networks (Jin et al., 2023). These systems consist of nodes that contain rich text information such as paper abstracts, product reviews, and tweets. These nodes form meaningful graphs, such as citation graphs, common purchasing graphs, and communication graphs. In this scenario, text and graph structure are not two independent entities. Edges connect the textual information of various nodes and they collectively form the dataset. Therefore, jointly utilizing all available text and graph data is crucial for handling variable downstream tasks.

When it comes to modeling text-attributed graphs, researchers typically focus on two primary aspects: one revolves around modeling the graph structure, while the other is centered on the text information contained within the graph. The first part, graph models, involves using techniques like Graph Neural Networks (GNNs) (Wu et al., 2019; Veličković et al., 2018; Zaknich, 1998; Huang et al., 2022; Hamilton et al., 2017) to refine the root node by aggregating information from neighboring nodes. On the other hand, word representation methods typically include bag-of-words vectors, fixed embeddings, and pre-trained large language models (Pennington et al., 2014; Devlin et al., 2019a). However, a disadvantage of this framework is that it treats text-level features and graph structure as separate entities, which cannot simultaneously consider information from both sides.

Consequently, achieving accurate and comprehensive analysis in these systems necessitates a thorough understanding not only of the semantic meaning of the nodes but also of the semantic meaning of their connected neighbors. Focusing solely on one side of the characteristics can easily result in biased predictions due to the omission of certain inherent information present in the original text and graph structure. To quantitatively define this problem, in Figure 1, we illustrate a case where **text bias** occurs when the graph model accurately predicts node labels, while the text model fails to do so. In the Amazon dataset, this bias accounts for approximately 9% of cases, where items possess clear neighbors and potentially misleading descriptions. Conversely, **graph bias** arises when the text model comprehends the semantic meaning of the text, but the graph model is incorrect due to the semantic-

[*] Chuang Zhou and Jiahe Du are co-first authors. Both authors contributed equally to this research.

[†] Xiao Huang is the corresponding author.

Proceedings of the 31st International Conference on Computational Linguistics, pages 10865–10876
January 19–24, 2025.

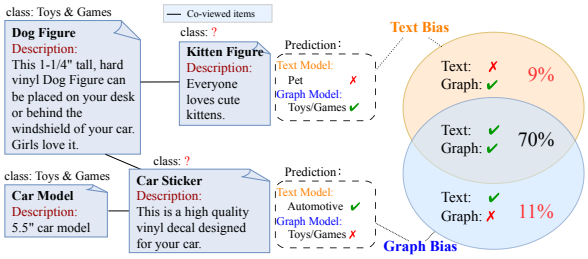

Figure 1: a) A demonstration of text bias and graph bias in the Amazon dataset. For the item "Kitten Figure", the graph model accurately assigns the label as "Toys/Games," while the text models are influenced by the misleading description. Conversely, for the item "car sticker," the graph model is influenced by its toy-related connections, whereas the text model provides the correct answer. b) The Amazon dataset exhibited text bias (9%) and graph bias (11%) ratios.

poor bag-of-words representation or influenced by misleading neighbors. Graph bias influences 11% of instances in the Amazon dataset given the data.

To tackle the aforementioned issues, it is crucial to establish a coupled interaction between embeddings obtained from different modules. However, learning these data relies on different sources of information. Graph models leverage network structure while text models concentrate on textual content. Another challenge is that real-life texts are not always labeled. Manual labeling is costly, so only a few graph nodes are annotated. Many unlabeled features may limit the utilization of traditional fine-tuning methods. Accordingly, we propose the Coupled Text and Graph Learning (CTGL) framework. Our framework introduces the Coupled Text & Graph Augmentation technique, which augments nodes' text information by incorporating textual data from graph-related or graph-unrelated nodes. Our contributions are summarized as:

- We have noticed that existing models tend to focus more on either graphs or text and formally define the bias towards each side.
- We propose a customized text augmentation technique that takes into account both textual semantic meanings and graph structure, guiding the text model to differentiate between similarities and differences from unlabeled nodes.
- By employing a shared augmentation strategy for text and graph contrastive learning, CTGL effectively bridges the gap between graph and text models, mitigating both forms of bias.
- Comprehensive experiments on three public datasets consistently demonstrate strong classification performance, with detailed studies confirming the effectiveness of each module.

2 Preliminaries

Notations. A text-attributed graph is a data structure that organizes information through nodes connected by edges, where each node is described by text and a label. It can be defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{S})$, where $\mathcal{S}$ represents text attributes for each node. $\mathcal{V}$ denotes the set of nodes, $\mathcal{E}$ denotes edges between text nodes and $\mathcal{N}_v^{(l)}$ denotes the l-hop neighbors of node v. Ground truth labels for a given text-attributed graph are denoted as $\boldsymbol{Y} = \{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_{|S|}\}$, where $|S|$ is the size of the text-attributed nodes. In real-world scenarios, only a small portion of the labels may be accessible.

3 Methodology

In this section, we present our CTGL framework for text-attributed graph learning. We begin by introducing the Coupled Text & Graph Augmentation (CTGA) technique and our framework. We then describe the Text Contrastive Learning process, which aims to enhance the text model's understanding of graph structure. Lastly, we propose the Graph Contrastive Learning approach, which incorporates text information during graph learning.

3.1 Overall Framework

As discussed in the preliminaries, language models often ignore graph structure, while current graph models tend to overlook the inclusion of semantic information in the text content. Integrating the two modalities presents a significant challenge due to two disparities: **(1) Data Sources**: Language models use semantic text for training, while graph models draw on the graph's structure. **(2) Embedding Space**: Language models generate embeddings directly from textual attributes, while graph models aggregate information obtained from neighboring nodes to refine the root node.

Coupled Training Framework. We propose a Coupled Text Graph Learning (CTGL) approach to address the disparities mentioned above. Our approach enables seamless interactions between text and graph models, without interrupting their respective training procedures. To achieve this, we introduce a graph-based loss to the text model and a text-based loss to the graph model. This strategy promotes the exchange of information between the two models, improving the overall learning process. The text learning loss is denoted as $\mathcal{L}_{\mathcal{T}}$, and the

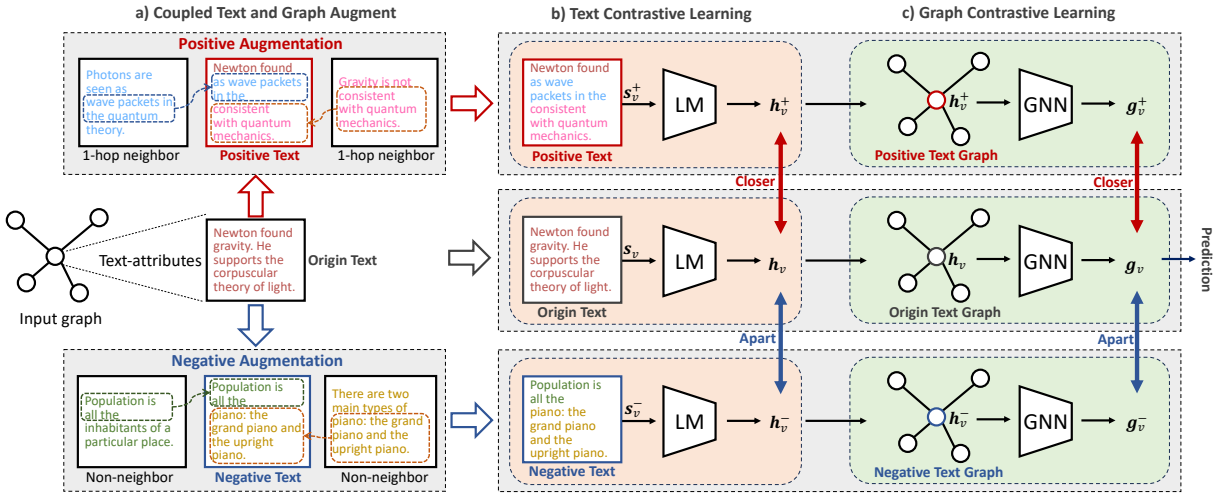

Figure 2: The proposed CTGL consists of three main parts. Subfigure a) illustrates the Coupled Text and Graph Augmentation mechanism and showcases how positive and negative views are generated. Subfigures b) and c) depict the structure of the coupled Text Contrastive Learning and Graph Contrastive Learning module.

graph learning loss as $\mathcal{L}_{\mathcal{G}}$, which are defined as,

$$\mathcal{L}_{\mathcal{T}} = \mathcal{L}_{\mathcal{T}}^{origin} + \lambda \mathcal{L}_{\mathcal{T}}^{couple}, \quad (1)$$

$$\mathcal{L}_{\mathcal{G}} = \mathcal{L}_{\mathcal{G}}^{origin} + \gamma \mathcal{L}_{\mathcal{G}}^{couple}, \quad (2)$$

where $\mathcal{L}_{\mathcal{T}}^{origin}$ and $\mathcal{L}_{\mathcal{G}}^{origin}$ denote the individual loss for the text and graph models, respectively, while $\mathcal{L}_{\mathcal{T}}^{couple}$ and $\mathcal{L}_{\mathcal{G}}^{couple}$ represent the coupled text and graph loss for each model.

Coupled Contrastive Learning. In the coupled contrastive learning phase, our goal is to configure the text model to generate analogous outputs for graph-related augmentations, and dissimilar outputs for graph-unrelated augmentations. Furthermore, we strive for the graph neural network to produce similar outputs for semantically related augmentations and disparate outputs for semantically unrelated augmentations.

We construct three views for both the text and graph models: the origin view, the positive view, and the negative view. The origin view computes the embeddings of the text and graph models by default. The positive view generates embeddings from positively sampled text and graph, whereas the negative mode does so from negatively sampled text and graph. Considering our goal of pulling the origin and positive embeddings closer, and pushing the origin and negative embeddings apart, the coupled text and graph losses can be subdivided as

$$\mathcal{L}_{\mathcal{T}}^{couple} = -sim(\boldsymbol{E}_{\mathcal{T}}, \boldsymbol{E}_{\mathcal{T}}^{+}) + sim(\boldsymbol{E}_{\mathcal{T}}, \boldsymbol{E}_{\mathcal{T}}^{-}), \quad (3)$$

$$\mathcal{L}_{\mathcal{G}}^{couple} = -sim(\boldsymbol{E}_{\mathcal{G}}, \boldsymbol{E}_{\mathcal{G}}^{+}) + sim(\boldsymbol{E}_{\mathcal{G}}, \boldsymbol{E}_{\mathcal{G}}^{-}). \quad (4)$$

In this context, $\boldsymbol{E}_{\mathcal{T}}$ and $\boldsymbol{E}_{\mathcal{G}}$ represent the embeddings of the text and graph model in the origin view, respectively. Similarly, $\boldsymbol{E}_{\mathcal{T}}^{+}$ and $\boldsymbol{E}_{\mathcal{G}}^{+}$ denote the embeddings in the positive view, while $\boldsymbol{E}_{\mathcal{T}}^{-}$ and $\boldsymbol{E}_{\mathcal{G}}^{-}$ denote those in the negative view. In the following subsections, we elucidate the process of coupled text and graph augmentation and the method to compute the embeddings.

3.2 Coupled Text and Graph Augmentation

At the heart of coupled contrastive learning is the design of Coupled Text and Graph Augmentations (CTGA), fostering simultaneous contrastive learning for text and graph models. Traditional augmentation strategies typically focus on single modal data augmentation, such as text or graph augmentation, for instance, modifying text or randomly dropping nodes or edges in graphs (Zhu et al., 2021). However, this approach isolates the text model from the graph model, thereby preventing the exchange of knowledge in between.

To address this, the design of CTGA accounts for the distinctive properties of text and graph models simultaneously. In the text modality, where the text represents the attributes of a given node, the neighbors have a closer relationship with the node than non-neighbors. In contrast, graph models consolidate neighboring text information to enhance the root node's text data. By holistically considering both modalities, we find that a node's final representation is profoundly influenced by its text attributes in both the text model and graph model. Therefore, we propose positive and negative views.

Positive views. The concept of positive augmentation refers to either text-based or graph-based augmentation. In the context of a text model, we aim for the original node to have an embedding similar to that of its neighbors. For the graph model, we aspire to achieve similar embeddings even when replacing the original node's text with the neighbors' text. Given that first-order neighbors may not inherently capture semantic similarity or class membership, we employ cosine similarity to devise a similarity-based sampling to select chunks from first-order neighbors. We start by using sentence-transformers to create representations of all nodes and then apply softmax to normalize the probabilities given pairwise similarity. Utilizing similarity for sampling allows for more precise control over the sample selection process, ensuring that the selected samples are more relevant or possess certain characteristics. Explicitly, for a specific node, we randomly select k sentences from its 1-hop neighbors and use these to replace the original text. Thus, the positive view can be further defined as follows,

$$\begin{aligned} \boldsymbol{s}_v^+ &= \{w_1, w_2, ..., w_k\}; \\ \{w_i \in \boldsymbol{s}_u | u \in \{\mathcal{N}_v^{(0)} \cup \mathcal{N}_v^{(1)}\}\}, \\ P(u|v) &= \frac{e^{-d(v,u)}}{\sum_{u'} e^{-d(v,u')}}, \end{aligned} \tag{5}$$

where w_i refers to the word sub-sequence drawn from neighbors, $\mathcal{N}_v^{(0)}$ and $\mathcal{N}_v^{(1)}$ denote the root node and the 1-hop neighbors, d denotes the min-max normalized cosine-similarity function.

Under these circumstances, positive samples are composed of text from neighboring nodes, which are likely to possess the same label and related text attributes. As such, the text model tends to inherently produce similar embeddings for the positively augmented text attribute. For the graph model, despite alterations to the root node's text attribute, the final embeddings produced by the graph neural network remain similar as the modifications are sourced from neighboring texts.

Negative views. Negative samples are built to differentiate content from spatially distant nodes.

$$\begin{aligned} \boldsymbol{s}_v^- &= \{w_1, w_2, ..., w_k\}; \\ \{w_i \in \boldsymbol{s}_u | u \notin \{\mathcal{N}_v^{(1)} \cup \mathcal{N}_v^{(2)}\}\}. \end{aligned} \tag{6}$$

where w_i refers to the word sub-sequence drawn from non-neighbors that do not belong to the 1-hop or 2-hop neighborhood of the root node, and $\mathcal{N}_v^{(1)}$ and $\mathcal{N}_v^{(2)}$ denote the 1-hop and 2-hop neighboring nodes, respectively.

In the text model, these negative samples consist of text from non-neighboring nodes, which are likely to have different labels and unrelated text attributes. Thus, the text model is expected to generate dissimilar embeddings for the negative samples. Meanwhile, for the graph model, modifying the root node's text attribute with non-neighboring nodes results in dissimilar embeddings, as the graph neural network discerns the distinction between the root node and its non-neighbors.

3.3 Text Contrastive Learning

The Text Contrastive Learning module learning consists of two integral components: 1) Text Embedding Learning, and 2) Text Contrastive Learning. The first part utilizes the 'cls token' of the tokenized sequence to effectively embed the original, positive, and negative text. By denoting the embedding function as $f_{cls}(\cdot)$, given three different views, their corresponding embedding of node v is $\boldsymbol{h}_v = f_{cls}(\boldsymbol{s}_v)$, $\boldsymbol{h}_v^+ = f_{cls}(\boldsymbol{s}_v^+)$ and $\boldsymbol{h}_v^- = f_{cls}(\boldsymbol{s}_v^-)$. The second part introduces a method of integration with the CTGA, aiming to ensure the model generates similar embeddings for graph-related augmentations while providing distinct outputs for graph-unrelated augmentations.

Text Classification Loss. As stated in Eq. (1), the complete text loss is made up of both the original classification loss and a weighted coupled loss. In terms of the original loss, we employ the cross-entropy classification loss. Given $\hat{\boldsymbol{y}}_v^{(t)} = f_{cls}^{(t)}(\boldsymbol{h}_v)$, The single loss for the text model is shown as,

$$\mathcal{L}_{\mathcal{T}}^{origin} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{K} \boldsymbol{y}_{i,c} \cdot \log \hat{\boldsymbol{y}}_{i,c}^{(t)}. \tag{7}$$

Text Contrastive Loss. The contrastive loss comprises two components, i.e., positive and negative view contrasts, following the description in Eq. (3). The former aims to maximize agreements between original texts and their positive views, while the latter is intended to increase the dissimilarity between the node vector $\boldsymbol{h}_v$ and its negative sample $\boldsymbol{h}_v^-$. only the node representation $\boldsymbol{h}_v$ and its corresponding augmentation $\boldsymbol{h}_v^+$ serve as the positive pair while any other combination within the batch is negatively treated, $sim(\boldsymbol{E}_{\mathcal{T}}, \boldsymbol{E}_{\mathcal{T}}^+)$ is defined as:

$$\mathcal{L}_{\mathcal{T}}^{pos} = \frac{1}{|\mathcal{V}|} \sum \log \frac{\exp(\phi(\boldsymbol{h}_v, \boldsymbol{h}_v^+)/\tau)}{\sum_{k=1}^{|\mathcal{V}|} \exp(\frac{\phi(\boldsymbol{h}_v, \boldsymbol{h}_v^+)}{\tau})}, \tag{8}$$

where $\phi(\boldsymbol{a}, \boldsymbol{b})$ function computes the cosine similarity between the vector $\boldsymbol{a}$ and $\boldsymbol{b}$. τ is the temperature hyperparameter. Additionally, the negative pair-wise loss $sim(\boldsymbol{E}_{\mathcal{T}}, \boldsymbol{E}_{\mathcal{T}}^{-})$ is defined as follows:

$$\mathcal{L}_{\mathcal{T}}^{neg} = \frac{1}{|\mathcal{V}|} \sum_{v=1}^{|\mathcal{V}|} \phi(\boldsymbol{h}_v, \boldsymbol{h}_v^{-}). \tag{9}$$

3.4 Graph Contrastive Learning

The Graph Contrastive Learning module comprises two key components: 1) Graph Embedding Learning and 2) Graph Contrastive Learning. The former generates graph model embeddings for the original, positive, and negative text attribute graphs, while the latter integrates with the CTGA to ensure the model generates similar embeddings for graph-related augmentations and distinct outputs for graph-unrelated augmentations.

Graph Embedding Learning. We compute the graph embedding for the origin, positive, and negative views separately. In the positive view, the input embedding of the root node is the positive embedding, as depicted in Figure 2 (c). Likewise, in the negative view, the input embedding of the root node is the negative embedding. Both positive and negative embeddings pass through the graph propagation layers first, with further details available in the appendix A.2.

Graph Module Loss. Similarly, the overall graph loss contains the original classification loss and contrast loss. As stated in Eq. (10), we aim to minimize the cross-entropy loss between the ground truth $y_{i,c}$ and the prediction $\hat{\boldsymbol{y}}_v^{(g)} = f_{cls}^{(g)}(\boldsymbol{g}_v^{(L)})$. In line with the text-level contrastive loss, as shown in Eq. (4), we denote the similarity between origin and positive embeddings as $sim(\boldsymbol{E}_{\mathcal{G}}, \boldsymbol{E}_{\mathcal{G}}^{+})$, as well as the similarity between origin and negative embeddings as $sim(\boldsymbol{E}_{\mathcal{G}}, \boldsymbol{E}_{\mathcal{G}}^{-})$. Our approach involves the development of batch-wise and pairwise contrastive loss, as depicted below:

$$\begin{aligned}
\mathcal{L}_{\mathcal{G}}^{ori} &= -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{K} y_{i,c} \cdot \log \hat{\boldsymbol{y}}_{i,c}^{(g)}. \\
\mathcal{L}_{\mathcal{G}}^{pos} &= \frac{1}{|\mathcal{V}|} \sum \log \frac{\exp(\phi(\boldsymbol{g}_v, \boldsymbol{g}_v^{+})/\tau)}{\sum_{k=1}^{|\mathcal{V}|} \exp(\frac{\phi(\boldsymbol{g}_v, \boldsymbol{g}_v^{+})}{\tau})}, \\
\mathcal{L}_{\mathcal{G}}^{neg} &= \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \phi(\boldsymbol{g}_v, \boldsymbol{g}_v^{-}).
\end{aligned} \tag{10}$$

4 Experiments

We conduct comprehensive experiments on three datasets across different proportions of labels. Our study addresses three research questions: (1) Can CTGL achieve superior prediction performance? (2) Can each module effectively align text-level and graph-level features and how do they perform compared with other baselines? (3) Is there a quantitative way to define text and graph bias and does our framework effectively alleviate such biases? (4) How can we intuitively understand embedding spaces possessed by different modalities?

4.1 Experimental Settings

Datasets. We assess the performance of CTGL using the following three datasets: ACM, Wikipedia, and Amazon. They are about academic papers, article categorization, and commercial products. For each dataset, we divide the labels into training, validation, and testing sets using three different proportions. The division of the training, validation, and testing datasets follows the settings used in previous studies (Zhang et al., 2022; Zhu et al., 2021). The detailed statistics of all three datasets are shown in table 2.

Datasets	#nodes	#edges	#classes
ACM	48,579	193,034	9
Wiki	36,501	1,190,369	10
Amazon	50,000	632,802	7

Table 2: Statistics of datasets in our experiment.

Evaluation Metrics. Main experiments are evaluated based on the prediction accuracy of the testing set. The corresponding standard deviation quantifies the variations observed across different folds and runs of the experiment. In prior discussions, we introduced the definitions of text and graph bias. Here we intend to mathematically define these concepts with conditional probability to respond **Q3**, which are shown as follows:

$$Bias_T = \mathcal{P}(\, f_G(\mathcal{G}, \mathcal{S}) = \boldsymbol{y} |\, f(\mathcal{G}, \mathcal{S}) \neq \boldsymbol{y}), \tag{11}$$

$$Bias_G = \mathcal{P}(\, f_L(\mathcal{G}, \mathcal{S}) = \boldsymbol{y} |\, f(\mathcal{G}, \mathcal{S}) \neq \boldsymbol{y}), \tag{12}$$

where f represents the target model to be evaluated and $\mathcal{P}$ is the probability. In Eq (11), f is chosen in f_{LM} and f_{CTGL} to measure their text bias, while in Eq (12) f is chosen in f_{GNN} and f_{CTGL} to measure their graph bias. The text bias refers to the

Emb	GNN	ACM 1%	ACM 5%	ACM 10%	Wiki 1%	Wiki 5%	Wiki 10%	Amazon 1%	Amazon 5%	Amazon 10%
BoW	GCN	$.514 \pm .023$	$.571 \pm .016$	$.596 \pm .010$	$.468 \pm .038$	$.547 \pm .011$	$.563 \pm .005$	$.745 \pm .015$	$.810 \pm .010$	$.833 \pm .006$
	SAGE	$.446 \pm .018$	$.516 \pm .009$	$.544 \pm .006$	$.441 \pm .024$	$.517 \pm .008$	$.538 \pm .009$	$.670 \pm .009$	$.750 \pm .006$	$.780 \pm .004$
	GATv2	$.504 \pm .019$	$.572 \pm .012$	$.599 \pm .012$	$.441 \pm .044$	$.535 \pm .017$	$.557 \pm .013$	$.690 \pm .015$	$.775 \pm .008$	$.807 \pm .006$
MPAD	GCN	$.603 \pm .017$	$.693 \pm .010$	$.716 \pm .006$	$.545 \pm .020$	$.601 \pm .009$	$.619 \pm .006$	$.813 \pm .018$	$.874 \pm .009$	$.891 \pm .005$
	SAGE	$.585 \pm .025$	$.663 \pm .020$	$.693 \pm .019$	$.554 \pm .024$	$.619 \pm .015$	$.638 \pm .005$	$.806 \pm .010$	$.868 \pm .007$	$.886 \pm .007$
	GATv2	$.623 \pm .013$	$.704 \pm .009$	$.731 \pm .007$	$.555 \pm .023$	$.633 \pm .009$	$.667 \pm .011$	$.806 \pm .017$	$.871 \pm .007$	$.894 \pm .003$
LM-fix	GCN	$.574 \pm .047$	$.594 \pm .079$	$.607 \pm .073$	$.533 \pm .025$	$.593 \pm .010$	$.603 \pm .009$	$.799 \pm .031$	$.852 \pm .010$	$.863 \pm .042$
	SAGE	$.562 \pm .031$	$.615 \pm .019$	$.626 \pm .023$	$.539 \pm .053$	$.578 \pm .108$	$.634 \pm .014$	$.736 \pm .119$	$.807 \pm .122$	$.833 \pm .085$
	GATv2	$.569 \pm .075$	$.618 \pm .101$	$.657 \pm .075$	$.550 \pm .027$	$.623 \pm .016$	$.655 \pm .010$	$.790 \pm .038$	$.857 \pm .008$	$.875 \pm .006$
LM-tune	GCN	$.642 \pm .011$	$.716 \pm .004$	$.734 \pm .003$	$.569 \pm .016$	$.636 \pm .009$	$.652 \pm .004$	$.844 \pm .018$	$.894 \pm .006$	$.910 \pm .003$
	GATv2	$.614 \pm .035$	$.714 \pm .011$	$.734 \pm .014$	$.562 \pm .028$	$.656 \pm .016$	$.689 \pm .010$	$.831 \pm .009$	$.886 \pm .011$	$.905 \pm .005$
	SAGE	$.556 \pm .112$	$.671 \pm .049$	$.691 \pm .071$	$.543 \pm .050$	$.645 \pm .009$	$.663 \pm .012$	$.810 \pm .024$	$.879 \pm .006$	$.898 \pm .004$
GLEM	GCN	$.559 \pm .057$	$.692 \pm .040$	$.725 \pm .010$	$.489 \pm .045$	$.635 \pm .018$	$.659 \pm .017$	$.772 \pm .032$	$.872 \pm .013$	$.894 \pm .007$
	SAGE	$.514 \pm .091$	$.710 \pm .009$	$.728 \pm .012$	$.490 \pm .051$	$.648 \pm .014$	$.672 \pm .011$	$.764 \pm .024$	$.872 \pm .009$	$.893 \pm .008$
	GATv2	$.518 \pm .087$	$.698 \pm .011$	$.727 \pm .011$	$.488 \pm .042$	$.651 \pm .012$	$.677 \pm .007$	$.761 \pm .027$	$.872 \pm .009$	$.888 \pm .007$
GraphFormers		$.611 \pm .014$	$.669 \pm .009$	$.677 \pm .001$	$.373 \pm .023$	$.500 \pm .008$	$.567 \pm .004$	$.730 \pm .011$	$.807 \pm .004$	$.832 \pm .001$
CTGL		$\mathbf{.668 \pm .016}$	$\mathbf{.727 \pm .008}$	$\mathbf{.748 \pm .003}$	$\mathbf{.583 \pm .020}$	$\mathbf{.668 \pm .011}$	$\mathbf{.698 \pm .005}$	$\mathbf{.861 \pm .010}$	$\mathbf{.907 \pm .002}$	$\mathbf{.921 \pm .002}$

Table 1: Comparison of baselines and our proposed framework across three public datasets.

conditional probability of misclassification caused by the target model f while the GNN (f_G) gives the right prediction and the same for the graph bias while the language model (f_L) gives the right prediction. Eq (11) and Eq (12) can eliminate the influence of the target model's overall accuracy, because conditional probability only counts the accuracy of LM or GNN among the target model's misclassification cases.

Implementation Details. A computer with 48 cores of 2.4 GHz Intel Xeon Silver 4214R CPU, 376 GB of RAM, and six NVidia GeForce RTX 3090 GPUs were used for all of the trials. We use the Adam optimizer with a learning rate equal to 0.001 and apply early stopping by observing the accuracy of the validation set. The hyperparameters λ and γ are tuned using a grid search and the best hyperparameters are chosen for each dataset. For fairness, all baselines follow the same design as descriptions in their articles.

Baselines. As our study focuses on integrating the corpus proceeding with the graph network, we adopt various popular approaches in text-embedding modules and GNN encoders. We made a cross combination and all baselines include GCN (Kipf and Welling, 2017), GraphSAGE (Hamilton et al., 2017), GATv2 (Brody et al., 2022), BoW (Zhang et al., 2010), MPAD (Nikolentzos et al., 2020), Language Model (LM-fix) (Yasunaga et al., 2022), Fine-tuning a language model (LM-tune), GLEM framework (Zhao et al., 2023) and GraphFormers (Yang et al., 2021).

4.2 Main Result (Q1)

The comparison of prediction performance across different proportions of labels on three datasets between CTGL and other baseline methods is presented in Table 1. To evaluate the effectiveness of CTGL, we conduct extensive experiments utilizing multiple text-embedding approaches and graph neural networks. The best result for each baseline group has been highlighted by underlying. It is shown that CTGL surpasses all baselines in terms of overall accuracy in classifying nodes across all datasets. This achievement is credited to the utilization of textual and graph learning. The shared augmentations derived from the corpus and topological structure collectively enhance the performance by improving the unity of the framework.

4.3 Effectiveness of TCL and GCL (Q2)

Effectiveness of TCL. To assess the effectiveness of the neighbor-based contrastive learning mechanism in improving text representation quality, we compare it with other methods such as bag-of-words, MPAD, fixed LM, and fine-tuned LM. To ensure fairness, we employ GATv2 as the graph encoder for all methods to generate classification results. As illustrated in Figure 3, TCL consistently surpasses other word embeddings in performance and efficiency. It is credited to our module aligning the corpus with the characteristics of the graph structure. Also, our data augmentation leverages textual information from unlabeled nodes, and the spatial relationship between documents mitigates the effects of text bias.

Effectiveness of GCL. We compare our GCL module with four classical contrastive-learning-based GNNs, namely COSTA (Zhang et al., 2022), GRAND (Chamberlain et al., 2021), GCA (Zhu et al., 2021), and DGI (Veličković et al., 2019). Traditional graph contrastive learning methods mostly rely on data augmentation techniques such as feature shuffling and edge dropping, focusing on either feature or structure perspective. In contrast, our module specifically targets text attributes-based augmentation, aiming to reduce graph bias resulting from node propagation. The results demonstrate that GCL achieves the best performance, highlighting the effectiveness of our tailored semantic augmentation in enhancing message propagation on graphs.

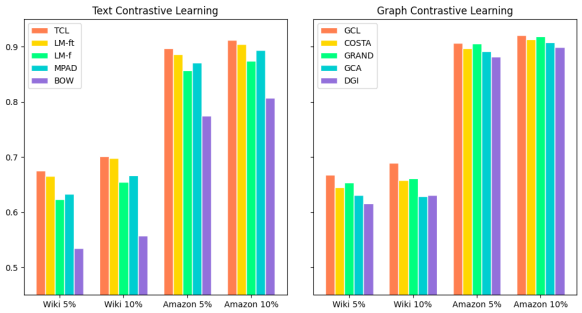

Figure 3: The effectiveness of TCL and GCL.

4.4 Text and Graph Bias Analysis (Q3)

As stated in Section 4.1, text bias refers to situations where the model fails to make accurate predictions while the GNN framework succeeds, and vice versa for graph bias. As shown in Table 3, we assess the performance of the GNN and LM in comparison with our own framework. The statistics reveal a significant inconsistency between the GNN and LM. In particular, the ACM dataset exemplifies a high degree of graph bias, whereby nearly half of the incorrect predictions made by the graph encoder can be rectified by language models. Text bias, although less severe, still exceeds 30%. In contrast, the CTGL gives a remarkable reduction in both text and graph bias. This conclusion is further supported by the results obtained from the other two datasets. In summary, the CTGL not only enhances overall accuracy but also effectively aligns corpus and graph information.

4.5 Visualization of Embedding Space (Q4)

We have previously mentioned that Graph models and Language models tend to map features into different vector spaces. Here, we attempt to visually observe the differences between them using the T-SNE method. Figure 4 displays the distributions of embeddings generated by four methods: Bag-of-words, Graph neural networks, language models, and our CTGL framework. Due to its sparsity and high dimensionality, BOW embeddings do not naturally provide good discrimination among different classes. The GNN encoder maps them to a lower and denser vector space, which helps alleviate some misclassification issues, although some incorrect instances still occur. Language models effectively distinguish texts of multiple categories; however, the representation of points tends to be closely clustered together, indicating that the embedding space's potential is not fully utilized. In contrast, our model demonstrates a noticeable distinction between chunks, with clear distances between them. The categories of the clusters are concentrated, showing our framework has learned a superior representation space.

Bias Type	Model	ACM	Wiki	Amazon
Graph Bias	GNN	0.512	0.488	0.625
	CTGL	**0.254**	0.300	0.273
	Improv.	+50.4%	+38.5%	+56.3%
Text Bias	LM	0.311	0.320	0.570
	CTGL	**0.250**	**0.213**	**0.292**
	Improv.	+19.6%	+33.4%	+48.8%

Table 3: Results of text and graph bias on three datasets. A reduction in bias results in improved outcomes.

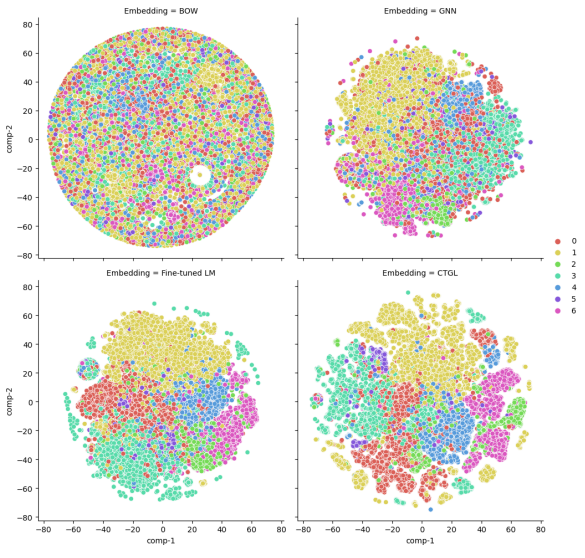

Figure 4: T-SNE visualization has demonstrated patterns of the representation space in which different embedding methods operate on the Amazon dataset.

4.6 Ablation Study

Module effect. In our study, we evaluate the effectiveness of the proposed debiasing semantic and network framework by comparing the performance of our CTGL model with its two variants: the Text-level and Graph-level Contrastive Learning modules. The weight of each component is controlled by λ and γ respectively, making this ablation study equivalent to testing the performance under different combinations of hyperparameters. Figure 5 presents comprehensive grid-search results, indicating that setting both hyperparameters to zero leads to relatively inferior accuracy. This suggests that removing any component results in a noticeable decrease in performance. The best result was attained when λ was configured at 1.5 and γ was set to 1. Furthermore, the presence of a flat plane at the top, highlighted in red, reflects the robustness of our model and indicates that the framework is not excessively sensitive to hyperparameters.

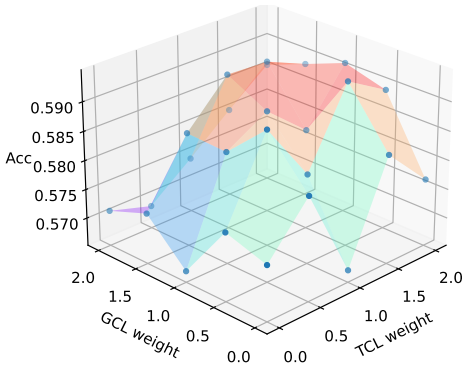

Figure 5: Hyperparameters grid-search results on Wiki.

Augmentation Method. We studied the performance of CTGL in four different scenarios, including whether to use negative samples and whether to use random sampling or distance normalization sampling when building positive samples. We conducted cross-experiments on these four scenarios, and the results are shown in the following table:

Effectiveness of Augmentation Method			
Sampling	Positive	Negative	Accuracy
Random	✓	×	.649
Random	✓	✓	.657
Normalized	✓	×	.663
Normalized	✓	✓	.668

Table 4: Comparisons across different augmentation methods. The accuracy is in % proportions.

5 Related Work

Text classification has long been a fundamental problem in the field of natural language processing. The common practice of utilizing a bag-of-words approach allows for straightforward construction based on word occurrence counts or Term Frequency-Inverse Document Frequency (TF-IDF) measures. Recent research findings suggest that appropriate pre-processing techniques can profoundly influence the final outcome (Zhou et al., 2023). Text-based features are often represented as one-hot vectors for node embeddings in many network analysis tasks (HaCohen-Kerner et al., 2020). Besides, alternative word embedding techniques have been introduced such as GloVe (Pennington et al., 2014). GPT (Brown et al., 2020; Dong et al., 2024a; Chen et al., 2024) and LLMs (Zhou et al., 2024; Dong et al., 2024b) are prominent models that leverage the Transformer architecture. There are some attempts to improve the performance of GNNs by utilizing large language models (Tan et al., 2023). GLEM adopts an Expectation Maximization algorithm to update two modules alternatively (Zhao et al., 2023). To encode text attributes along edges, the Edgeformers framework utilizes transformer layers (Jin et al., 2023). Recent studies on contrastive learning in TAG scenarios include Grenade, which leverages two self-supervised learning algorithms (Li et al., 2023). ConGraT (Brannon et al., 2023) aligns the representations of a language model and a GNN model in a shared space.

6 Conclusion

Coupled Text-attributed Graph Learning (CTGL) primarily addresses the problem of text-attributed node classification. Adopting the large language model leads to a better node representation than currently widely-used approaches such as Bag-of-words. To mitigate the impact of text and graph bias, the coupled contrastive learning mechanism builds a connection between these two domains, facilitating a more balanced interaction between text and graph learning. CTGL effectively enhances performance by jointly utilizing the graph structure and corpus information. Experiments on three datasets demonstrate its superiority and practicability in real-life tasks. Moreover, the ablation study highlights the necessity of each module. Visualization of different embedding spaces gives an intuitive understanding and quantitative case studies showcase the effectiveness of bias reduction.

Limitations

One limitation of this paper is that incorporating augmentation components into the framework inevitably requires additional computational resources. Consequently, the execution time of our framework is slightly longer compared to a direct fine-tuning pipeline. However, according to our experimental findings, the additional time required is entirely reasonable; the sampling process takes under 5 minutes for 100,000 nodes and can be performed offline using a CPU. Moreover, the overall training time for our framework can be finished within thirty minutes for the current dataset. Given the enhanced prediction accuracy, the associated cost is acceptable.

Ethics Statement

We all comply with the ACL Ethics Policy[1] in this study. We used one open-resource dataset that has been extensively used in prior research. For the other two datasets built by ourselves, the consumer information has been anonymized and privacy has been carefully protected.

Acknowledgments

The work described in this paper was fully supported by a grant from the Innovation and Technology Commission of the Hong Kong Special Administrative Region, China (Project No. GHP/391/22).

References

William Brannon, Wonjune Kang, Suyash Fulay, Hang Jiang, Brandon Roy, Deb Roy, and Jad Kabbara. 2023. Congrat: Self-supervised contrastive pretraining for joint graph and text embeddings. *arXiv preprint arXiv:2305.14321*.

Shaked Brody, Uri Alon, and Eran Yahav. 2022. How attentive are graph attention networks? In *International Conference on Learning Representations*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Benjamin Paul Chamberlain, James Rowbottom, Maria I. Gorinova, Stefan D Webb, Emanuele Rossi, and Michael M. Bronstein. 2021. GRAND: Graph neural diffusion. In *The Symbiosis of Deep Learning and Differential Equations*.

Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *International Conference on Learning Representations*.

Shengyuan Chen, Qinggang Zhang, Junnan Dong, Wen Hua, Qing Li, and Xiao Huang. 2024. Entity alignment with noisy annotations from large language models. *arXiv preprint arXiv:2405.16806*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Junnan Dong, Qinggang Zhang, Chuang Zhou, Hao Chen, Daochen Zha, and Xiao Huang. 2024a. Cost-efficient knowledge-based question answering with large language models. *arXiv preprint arXiv:2405.17337*.

Junnan Dong, Qinggang Zhang, Huachi Zhou, Daochen Zha, Pai Zheng, and Xiao Huang. 2024b. Modality-aware integration with large language models for knowledge-based visual question answering. *arXiv preprint arXiv:2402.12728*.

Yaakov HaCohen-Kerner, Daniel Miller, and Yair Yigal. 2020. The influence of preprocessing on text classification using a bag-of-words representation. *PloS one*, 15(5):e0232525.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 1025–1035.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, page 507–517.

Zhongyu Huang, Yingheng Wang, Chaozhuo Li, and Huiguang He. 2022. Going deeper into permutation-sensitive graph neural networks. In *International Conference on Machine Learning*, pages 9377–9409. PMLR.

[1] https://www.aclweb.org/portal/content/acl-code-ethics

Bowen Jin, Yu Zhang, Yu Meng, and Jiawei Han. 2023. Edgeformers: Graph-empowered transformers for representation learning on textual-edge networks. In *The Eleventh International Conference on Learning Representations*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Yichuan Li, Kaize Ding, and Kyumin Lee. 2023. Grenade: Graph-centric language model for self-supervised representation learning on text-attributed graphs. *arXiv preprint arXiv:2310.15109*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari.

Giannis Nikolentzos, Antoine Tixier, and Michalis Vazirgiannis. 2020. Message passing attention networks for document understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8544–8551.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pages 1532–1543.

Qiaoyu Tan, Xin Zhang, Jiahe Du, and Xiao Huang. 2023. Graph neural networks with non-recursive message passing. In *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 506–514. IEEE.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 990–998.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. In *International Conference on Learning Representations*.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871.

Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit S, Guangzhong Sun, and Xing Xie. 2021. Graphformers: GNN-nested transformers for representation learning on textual graph. In *Advances in Neural Information Processing Systems*.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2022. LinkBERT: Pretraining language models with document links. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8003–8016, Dublin, Ireland. Association for Computational Linguistics.

Anthony Zaknich. 1998. Introduction to the modified probabilistic neural network for general signal processing applications. *IEEE Transactions on Signal Processing*, 46(7):1980–1990.

Peiyan Zhang, Jiayan Guo, Chaozhuo Li, Yueqi Xie, Jae Boum Kim, Yan Zhang, Xing Xie, Haohan Wang, and Sunghun Kim. 2023. Efficiently leveraging multi-level user intent for session-based recommendation via atten-mixer network. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 168–176.

Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. 2022. Costa: Covariance-preserving feature augmentation for graph contrastive learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 2524–2534.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. 2010. Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics*, 1:43–52.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2023. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations*.

Chuang Zhou, Junnan Dong, Xiao Huang, Zirui Liu, Kaixiong Zhou, and Zhaozhuo Xu. 2024. Quest: Efficient extreme multi-label text classification with large language models on commodity hardware. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3929–3940.

Huachi Zhou, Hao Chen, Junnan Dong, Daochen Zha, Chuang Zhou, and Xiao Huang. 2023. Adaptive popularity debiasing aggregator for graph collaborative filtering.(2023). *A APPENDICES A*, 1.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, page 2069–2080.

A Appendix

A.1 Text Model

Traditional text models Traditional text classification models utilize vectors to represent text and classifiers to predict labels such as one-hot vectors, bag-of-words, and word embeddings. Then classifiers like MLP and RNNs (Mikolov et al., 2010) are used to tackle the embedding sequence to make the prediction. Overall, the predicted label for node v can be presented as:

$$\hat{\boldsymbol{h}}_v^{(t)} = f_{cls}(\{\boldsymbol{x}_{v,1}, \cdots, \boldsymbol{x}_{v,|\boldsymbol{s}_v|}\}), \quad (13)$$

where $\hat{\boldsymbol{y}}_v^{(t)}$ is the prediction result of the the text model, f_{cls} denotes the classification layer and $\boldsymbol{x}_{v,n}$ denotes the representation of the n-th textual entity in $\boldsymbol{s}_v$. $|\boldsymbol{s}_v|$ denotes the length of the sentence $\boldsymbol{s}_v$.

Pre-trained language models In contrast to conventional text models, BERT (Devlin et al., 2019b) utilizes a pre-trained, large-scale language model from extensive corpora. BERT comprises an embedding function f_{emb} and a predictive MLP layer $f^(t)_{cls}$. Subsequently, when presented with a text classification task, BERT dynamically generates the text's embedding and classifies it using the prediction layer. The complete prediction formula for the pre-trained language model is defined as follows,

$$\boldsymbol{h}_v = f_{emb}(\boldsymbol{s}_v), \;\; \hat{\boldsymbol{y}}_v^{(t)} = f_{cls}(\boldsymbol{h}_v), \quad (14)$$

where $\boldsymbol{h}_v$ denotes the pre-trained embedding of the text content $\boldsymbol{s}_v$. Despite achieving state-of-the-art performance in text classification tasks, pre-trained language models are unable to utilize or understand the graph structure present in text-attributed graphs.

A.2 Graph Neural Network

Graph Neural Networks (Chen et al., 2018; Hamilton et al., 2017) use recursive graph convolution layers to enhance the root node by aggregating information from its neighbors. However, the computational complexity of these layers increases exponentially (Zhang et al., 2023), so fixed embedding vectors (e.g. word embeddings) are often employed for modeling text content. The aggregation process of a single graph convolution layer can be expressed as:

$$\boldsymbol{g}_v^{(l+1)} = \sigma(\text{FC}_{\theta^{(l)}}(w_{v,v}^{(l)} \cdot \boldsymbol{g}_v^{(l)} + \sum_{u \in \mathcal{N}_v^{(1)}} w_{u,v}^{(l)} \cdot \boldsymbol{g}_u^{(l)})), \quad (15)$$

where $\text{FC}(\cdot)$ is the fully connect layer with parameter $\theta^{(l)}$, $w_{u,v}^{(l)}$ is the aggregation weight computed from neighbor u to root node v. Specifically, $\boldsymbol{g}_v^{(0)}$ is given by the input embedding vector.

Then the final prediction of the graph model can be stated as:

$$\hat{\boldsymbol{y}}_v^{(g)} = f_{cls}^{(g)}(\boldsymbol{g}_v^{(L)}), \quad (16)$$

where $f_{cls}^{(g)}$ denotes the classification function for the graph model, and L denotes the number of the graph convolution layer.

To provide the augmented graph convolution for positive views and negative views, we modify Eq.(15) as follows,

$$\begin{aligned} & \boldsymbol{g}_v^{(l+1),+} \quad (17) \\ = \; & \sigma(\text{FC}_{\theta^{(l),-}}(w^{(l),+}(\boldsymbol{g}_v^{(l),+}, \boldsymbol{g}_v^{(l),+}) \cdot \boldsymbol{g}_v^{(l),+} \\ & + \sum_{u \in \mathcal{N}_v^{(1)}} w^{(l),+}(\boldsymbol{g}_v^{(l),+}, \boldsymbol{g}_u^{(l)}) \cdot \boldsymbol{g}_u^{(l)})), \\ & \boldsymbol{g}_v^{(l+1),-} \quad (18) \\ = \; & \sigma(\text{FC}_{\theta^{(l),-}}(w^{(l),-}(\boldsymbol{g}_v^{(l),-}, \boldsymbol{g}_v^{(l),-}) \cdot \boldsymbol{g}_v^{(l),-} \\ & + \sum_{u \in \mathcal{N}_v^{(1)}} w^{(l),-}(\boldsymbol{g}_v^{(l),-}, \boldsymbol{g}_u^{(l)}) \cdot \boldsymbol{g}_u^{(l)})), \end{aligned}$$

where $\sigma(\cdot)$ is the nonlinear function, $\boldsymbol{g}_v^{(0),+} = \boldsymbol{h}_v^+$ and $\boldsymbol{g}_v^{(0),-} = \boldsymbol{h}_v^-$. Specifically, using the positive view as the example, the aggregation weight function $w^{(l),+}(\cdot,\cdot)$ can be computed as follows,

$$\begin{aligned} & r^{(l),+}(\boldsymbol{g}_v^{(l),+}, \boldsymbol{g}_u^{(l)}) \quad (19) \\ = \; & \alpha^{(l),+\top} \sigma(\boldsymbol{W}^{(l),+} \cdot [\boldsymbol{g}_v^{(l),+} \parallel \boldsymbol{g}_u^{(l)}], \\ & w^{(l),+}(\boldsymbol{g}_v^{(l),+}, \boldsymbol{g}_u^{(l)}) \quad (20) \\ = & \frac{\exp(r^{(l),+}(\boldsymbol{g}_v^{(l),+}, \boldsymbol{g}_u^{(l)}))}{\sum_{k \in \mathcal{N}_v^{(1)}} \exp\left(r^{(l),+}(\boldsymbol{g}_v^{(l),+}, \boldsymbol{g}_k^{(l)})\right)}, \end{aligned}$$

where $\parallel$ denotes the vector concatenation, $\alpha^{(l),+}$ and $\boldsymbol{W}^{(l),+}$ denote the weight vector and matrix for the weight computation for positive augmentations.

A.3 Dataset

Wikipedia The raw data consists of UTF-8 encoded text from Wikipedia articles[2]. We extract the main content of each article as document d_v, which includes hyperlinked words. A directed graph is constructed using the hyperlink relationships between articles. The categories mentioned in the *list of reference tables* are assigned as labels to the nodes.

[2]http://www.mattmahoney.net/dc/textdata

ACM This dataset uses 48,579 papers from the Association for Computing Machinery (ACM) as instances (Tang et al., 2008). The paper abstracts serve as the document d_v for the nodes, and a directed graph is constructed using the citation links. The instances are collected from nine distinct domains, such as Artificial Intelligence, Data Mining, and Machine Learning, which are employed as labels.

Amazon The dataset comprises product reviews and metadata from Amazon (He and McAuley, 2016). We construct the graph based on the browsing history, with each node v representing the textual description of the products denoted as s_v.

A.4 Baselines

Here are the introductions of each method:

- GCN (Kipf and Welling, 2017) aggregates information from neighboring nodes by summing over neighbors' representations.
- GraphSAGE (Hamilton et al., 2017) samples and aggregates features from the neighborhood for inductive graph learning.
- GATv2 (Brody et al., 2022) introduces a dynamic graph attention mechanism, leveraging attention layers to learn the weights of neighboring features.
- Bag of Words (BoW) (Zhang et al., 2010) describes the occurrence of words within a document and its size can be flexibly decided by the frequencies of different words.
- MPAD (Nikolentzos et al., 2020) represents corpus as networks based on word co-occurrence and applies a message-passing framework to draw the information from the graph.
- Language Model (LM-fix) (Yasunaga et al., 2022) is first trained on a large set of corpus and it can be adopted in different downstream tasks with fixed embeddings as text representations.
- Fine-tuning a language model (LM-tune) allows for training on target texts to make the model more adept at performing the specific task.
- GLEM framework (Zhao et al., 2023) iteratively updates the language model and graph neural network (GNN).