SCALE-FREE GRAPH-LANGUAGE MODELS

Jianglin Lu[1]* **Yixuan Liu**[2] **Yitian Zhang**[1] **Yun Fu**[1,3]
[1]Department of Electrical and Computer Engineering, Northeastern University
[2]Network Science Institute, Northeastern University
[3]Khoury College of Computer Science, Northeastern University

ABSTRACT

Graph-language models (GLMs) have demonstrated great potential in graph-based semi-supervised learning. A typical GLM consists of two key stages: graph generation and text embedding, which are usually implemented by inferring a latent graph and finetuning a language model (LM), respectively. However, the former often relies on artificial assumptions about the underlying edge distribution, while the latter requires extensive data annotations. To tackle these challenges, this paper introduces a novel GLM that integrates graph generation and text embedding within a unified framework. Specifically, for graph generation, we leverage an inherent characteristic of real edge distribution—the *scale-free property*—as a structural prior. We unexpectedly find that this natural property can be effectively approximated by a simple k-nearest neighbor (KNN) graph. For text embedding, we develop a graph-based pseudo-labeler that utilizes scale-free graphs to provide complementary supervision for improved LM finetuning. Extensive experiments on representative datasets validate our findings on the scale-free structural approximation of KNN graphs and demonstrate the effectiveness of integrating graph generation and text embedding with a real structural prior. Our code is available at https://github.com/Jianglin954/SFGL.

1 INTRODUCTION

Recently, graph-language models (GLMs) have been widely explored in graph-based semi-supervised classification on documents, especially for citation networks (Qin et al., 2023; Yu et al., 2025; Lu et al., 2023; He et al., 2024). When designing a GLM for classification, two key challenges arise: *graph generation*—how to generate a reasonable graph structure for the given documents, and *text embedding*—how to encode the textual sequences into meaningful semantic features. To address these problems, various GLMs have been proposed, which can be broadly categorized into latent graph inference (LGI) models and language-assisted graph (LAG) models.

LGI models focus on graph generation and typically rely on feature engineering approaches, such as bag-of-words (Harris, 1954), TF-IDF (Aizawa, 2003), and skip-gram (Mikolov et al., 2013), to encode textual sequences into shallow representations. These representations serve as the foundation for optimization objectives that jointly learn the underlying graph structure and node embeddings using graph neural networks (GNNs) (Franceschi et al., 2019; Fatemi et al., 2021; Lu et al., 2023; Kazi et al., 2023). *However, LGI models typically rely on artificial assumptions about the underlying edge distribution, which may not accurately reflect the real graph structure.*

On the other hand, LAG models assume a predefined graph structure and focus on enhancing text embedding through powerful language models (LMs), such as DeBERTa (He et al., 2021) and GPT (Ouyang et al., 2022). The core idea behind LAG models is to finetune LMs on the target datasets, thereby generating more informative text embeddings (Yu et al., 2025; Duan et al., 2023; He et al., 2024). *However, finetuning a pretrained LM typically requires target datasets with extensive annotations (Brown et al., 2020), which is often impractical for semi-supervised learning tasks. When sufficient annotations are unavailable, LM finetuning may become unreliable.*

In this paper, we propose a novel scale-free graph-language (SFGL) model that integrates graph generation and text embedding into a unified framework. The proposed SFGL framework addresses

*Corresponding author: `jianglinlu@outlook.com` the challenges of relying on specious edge distribution assumptions and mitigates the limitations of insufficient supervision in LM finetuning. This is accomplished by identifying an inherent structural nature of real networks for rational graph generation and incorporating a graph-based pseudo-labeler to enhance LM finetuning. Specifically, we investigate a fundamental characteristic of edge distribution in real networks: the *scale-free property* (Barabási & Bonabeau, 2003; Radicchi et al., 2011; Barabási & Pósfai, 2016), which is primarily characterized by a few highly connected hubs and numerous small nodes with few edges (see Fig. 1 for an example). The scale-free edge distribution is prevalent in real-world networks, particularly in citation networks, making it an ideal structural prior for graph generation. However, the dynamic and sequential nature of its formation process makes efficiently fitting this distribution highly challenging. Fortunately, we reveal that *a k-nearest neighbor (KNN) graph, constructed using cosine similarity as the distance metric and an appropriately chosen k, closely approximates a scale-free network*. We subsequently leverage this scale-free graph to develop a graph-based pseudo-labeler, which provides complementary supervision for enhancing LM finetuning. Consequently, the improved LMs can generate semantically enriched text embeddings. Note that our proposed SFGL model can be trained iteratively to further enhance performance. We conduct extensive experiments to validate our findings and highlight the potential of GLMs built on scale-free structures. Our key contributions are summarized as follows:

- We identify two key challenges in existing GLMs: artificial structural assumptions in graph generation and unreliable LM finetuning for text embedding. We propose addressing these challenges simultaneously by exploring a well-grounded graph structural prior.
- We leverage the scale-free edge distribution in real networks as our graph structural prior. Our empirical validation and analysis reveal that a KNN graph, constructed using cosine similarity with an appropriately chosen k, effectively approximates a scale-free network.
- To the best of our knowledge, the proposed SFGL is the first work to unify graph generation and text embedding within a GLM framework, highlighting the synergistic potential of GNNs and LMs under a scale-free structural prior.

2 Preliminaries

2.1 Problem Definition

Given a set of documents denoted as $\mathcal{D} = \{\mathcal{X}_L, \mathcal{X}_U, \mathcal{Y}_L\}$, where $\mathcal{X}_L = \{\mathbf{x}_i\}^m$ and $\mathcal{X}_U = \{\mathbf{x}_i\}^n$ represent the text sequences of m labeled and n unlabeled documents, respectively, and $\mathcal{Y}_L = \{\mathbf{y}_i\}^m$ is the label set for the labeled samples, with $m \ll n$ indicating a scarcity of labeled nodes compared to the abundance of unlabeled ones, we tackle a graph-based semi-supervised classification task that involves training a model on $\mathcal{D}$ to classify unlabeled documents $\mathcal{X}_U$. Here, we face two key problems: *graph generation*—how to generate a reasonable graph structure for the given documents, and *text embedding*—how to encode the textual sequences into meaningful semantic features.

For graph generation, the main challenge lies in identifying a reasonable structural prior that characterizes the underlying distribution of edges. The choice of this prior is critical, as it determines both the complexity of the learning model and the inductive bias introduced. We suggest that an effective structural prior should exhibit two key characteristics: *real*—it should reflect an inherent attribute of the data; and *simple*—it should allow for straightforward graph generation without requiring additional model training.

For text embedding, advanced LMs are preferred over traditional feature engineering approaches for their superior ability to capture semantic information from textual sequences of documents. The primary challenge here lies in improving LM finetuning in a semi-supervised setting with very limited labeled samples. We suggest addressing this challenge with a graph-based semi-supervised learning model that propagates supervised information from labeled to unlabeled samples through the graph structure. However, this brings us back to the central question: how to identify a reasonable structural prior for graph generation.

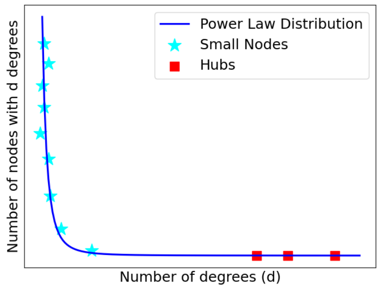


Figure 1: Illustration of a scale-free network, highlighting a few *hubs* and a large number of *small nodes*.

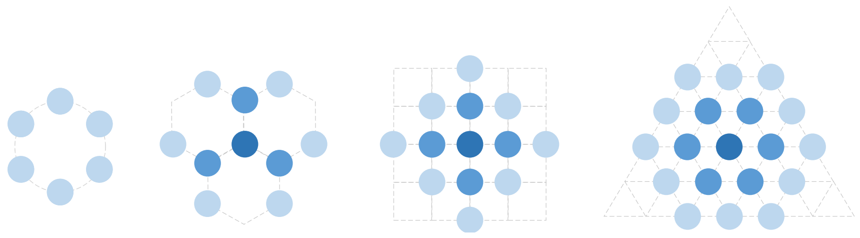


Figure 2: Uniform node distribution within structured arrangements causes consistent connectivity.

2.2 Scale-Free Networks

We investigate a fundamental structural nature of real-world networks: the *scale-free property* (Barabási & Albert, 1999; Barabási & Bonabeau, 2003). In essence, real citation networks exhibit scale-free characteristics (Redner, 1998; Hummon & Dereian, 1989; Radicchi et al., 2011). Their degree distribution $\mathbb{P}(\theta)$ follows a power law: $\mathbb{P}(\theta) \propto \theta^{-\alpha}$, where θ denotes the degree of a node and α is a scaling parameter of the distribution (Clauset et al., 2009). For illustration, Fig. 1 depicts the degree distribution of a scale-free network with $\alpha = 2.8$. As observed, the distribution is heavy-tailed, with a few highly connected nodes, known as *hubs*, and many *small nodes* with only a few edges (Barabási & Albert, 1999; Barabási & Pósfai, 2016). We suggest that leveraging the scale-free property for graph generation is a strong choice, as it reflects an inherent characteristic of citation networks and aligns with our *real* principle. The challenge lies in constructing a latent graph that can (approximately) form a scale-free network.

2.3 K-Nearest Neighbor Graphs

We begin by exploring KNN graphs, as their construction aligns with our *simple* principle. The construction process is straightforward: for each node, we identify its k-nearest neighbors using a specified distance metric based on node features and establish an edge between the node and each of its k nearest neighbors. The degree distribution of KNN graphs is primarily shaped by three key factors: ① *node distribution*, which is unknown a priori; ② *the choice of similarity metric*, which affects the probability of node connections; and ③ *the value of* k, which directly determines node degrees. Without detailed knowledge on these factors, characterizing the degree distribution of KNN graphs remains challenging. We first explore simple cases and analyze each factor independently.

In the absence of prior knowledge about the underlying node distribution, we analyze attribute-free nodes that are uniformly distributed across regular grid structures, including circles, hexagons, quadrilaterals, and triangles. As shown in Fig. 2, the subgraph formed by any node and its first-order neighbors exhibits characteristics of a random network (Barabási & Albert, 1999). Within a specified value of k, the connection probabilities between two nodes within the subgraph remain identical. Thus, building KNN graphs on such uniform distributions yields consistent node connectivity.

In KNN graphs, various distance metrics can be used to measure similarity between nodes, including Euclidean, Manhattan, and Cosine distances. Euclidean distance is sensitive to variations in feature scale and highly influenced by outliers. Manhattan distance is more robust to outliers but disregards the actual geometric relationships between data points. Cosine distance is particularly suitable for high-dimensional, sparse data, as it prioritizes vector direction over magnitude.

The value of k directly determines the magnitude of node degrees. As k increases, more edges are formed, leading to higher node degrees. In the extreme case where $k = m + n - 1$, the KNN graph becomes a complete graph, where every node is connected to all others and has the same degree.

3 Scale-Free Graph-Language Model

3.1 Scale-Free Graph Generation

Hypothesis & Analysis. In this paper, we argue that constructing a KNN graph that approximates a scale-free network is practically feasible. To support this, we first present the following hypothesis:

A k-nearest neighbor graph, when constructed on citation networks using cosine similarity as the distance metric and an appropriately chosen k, approximates a scale-free network.

To investigate the proposed hypothesis, we present the following definitions and propositions.

Definition 1. *Given an undirected graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges, the degree centrality of a node v is $C_D(v) = \theta(v)$, where $\theta(v)$ is the degree of v.*

Definition 2. *Given a directed graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges, the in-degree centrality and out-degree centrality of a node v are $C_{ID}(v) = \theta_i(v)$ and $C_{OD}(v) = \theta_o(v)$, where $\theta_i(v)$ and $\theta_o(v)$ are the in-degree and out-degree of v, respectively.*

Proposition 1. *For a scale-free network, assuming that there is at most one node v whose degree belongs to $[\theta_{max}, \infty)$, we have $\theta_{max} = \theta_{min}|\mathcal{V}|^{\frac{1}{\alpha-1}}$, where θ_{max} and θ_{min} refer to the maximum and minimal degree in the scale-free network.*

Proposition 2. *For a KNN graph, we have $\sum_{v\in\mathcal{V}} C_{ID}(v) = \sum_{v\in\mathcal{V}} C_D(v) - \sum_{v\in\mathcal{V}} C_{OD}(v) = 2|\mathcal{E}| - k|\mathcal{V}|$, where $|\mathcal{E}| \leq k|\mathcal{V}|$.*

Considering their inherent biases, KNN graphs exhibit significant similarities to scale-free networks, which we detail as follows. ① *Homophily:* citation networks exhibit a high degree of homophily, where connected nodes usually have a high probability of belonging to the same class. Notably, the KNN algorithm is also based on the homophily assumption that similar nodes exist in close proximity. ② *Directed:* citation networks are inherently directed since citations have a clear direction from the citing publication to the cited publication. Coincidentally, the KNN graph is also directed as node v being in the k-nearest neighbors of node u does not imply node u is in the k-nearest neighbors of node v. ③ *Upper bound of degree*: the size of hubs in both scale-free networks and KNN graphs is constrained. Proposition 1 indicates that the size of hubs in scale-free networks has an upper bound of $d_{min}|\mathcal{V}|^{\frac{1}{\alpha-1}}$. Note that the maximal in-degree in a KNN graph is also bounded, and the out-degree of each node is always fixed as k, as shown in proposition 2.

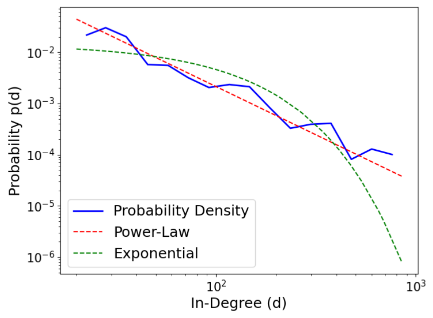
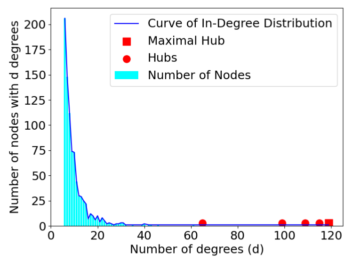


Figure 3: Curve fitting to the in-degree distribution of a KNN graph ($k = 5$) using Euclidean distance as the metric.

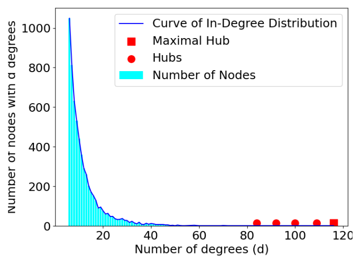
We discuss the underlying reasons and rationale for using "cosine similarity" and the term "approximates". Cosine similarity focuses on the orientation of feature vectors rather than their magnitude, aligning with the syntactic patterns of word semantics. Consequently, few hubs with high in-degrees will appear in KNN graphs if the feature space exhibits some common orientations. The term "approximates" is used because KNN graphs are constructed based on node features, which are influenced by the feature engineering approaches employed. After feature extraction, the node distribution may not accurately represent the original data distribution of citation networks.

Empirical Validation. To validate our hypothesis, we construct KNN graphs using cosine similarity with $k = 5$ and examine their degree distribution across multiple citation networks, including `Cora`, `Pubmed`, and `ogbn-arxiv` (see Appendix A.1 for details). Figure 4 presents the in-degree distribution of KNN graphs on these datasets. Our observations reveal that the generated KNN graphs contain massive small nodes alongside a few hubs, resulting in in-degree distributions that resemble those of scale-free networks as depicted in Fig. 1. Notably, the degree distributions exhibit slight variations across datasets, which can be attributed to differences in node distributions and the feature extraction methods used to construct feature vectors. To further investigate these degree distributions, we fit the in-degree distributions to exponential and power-law models—both characterized by heavy-tailed behavior. Figure 5 illustrates the fitting curves on log-log axes using logarithmically spaced bins. As shown, the in-degree distributions of KNN graphs align more closely with a power law, with estimated scaling exponents (α) of 3.3, 2.8, and 2.7 for `Cora`, `Pubmed`, and `ogbn-arxiv`, respectively. These empirical findings suggest that KNN graphs effectively capture the structural properties of citation networks, thus supporting our hypothesis.

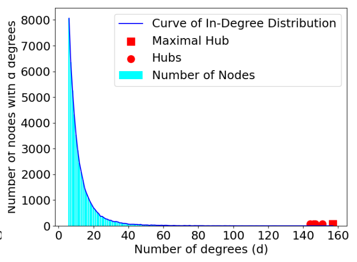
We further investigate the degree distribution of KNN graphs using Euclidean distance and present the fitting curves in Fig. 3. As shown, the resulting degree distribution does not fit well to a power-



(a) Cora

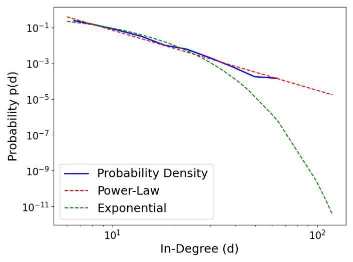


(b) Pubmed

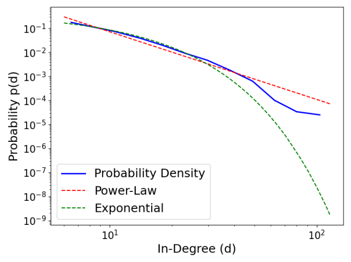


(c) ogbn-arxiv

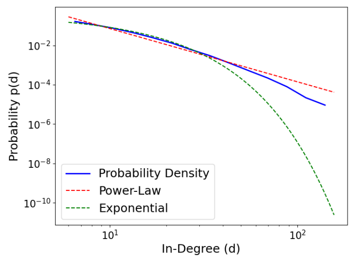
Figure 4: In-degree distribution of KNN graphs on different datasets, where the top-5 largest hubs are marked in red.



(a) Cora



(b) Pubmed



(c) ogbn-arxiv

Figure 5: Curve fitting to the in-degree distribution of KNN graphs on different datasets.

law model, indicating that Euclidean distances between vector points fail to accurately capture the similarities between textual features in citation networks. This is because Euclidean distance primarily emphasizes the magnitude of vectors in linear space, neglecting important elements such as the angular distribution or relative orientation of vectors, which are essential for capturing syntactic patterns and semantic relationships of textual attributes. More comparisons are provided in Sec. 4.3.

Graph Construction. Building on the above analysis and validation, we generate a scale-free KNN graph for the given documents $\mathcal{D}$. We begin by encoding the text attributes $\mathcal{X} = \{\mathcal{X}_L \cup \mathcal{X}_U\} = \{\mathbf{x}_i\}^{m+n}$ into hand-crafted feature representations using a text encoder $\texttt{Enc}(\cdot)$: $\mathbf{f}_i = \texttt{Enc}(\mathbf{x}_i)$, where $\mathbf{f}_i$ represents the extracted feature vector for $\mathbf{x}_i$, and $\texttt{Enc}(\cdot)$ can be either a traditional model or a language model. Next, we construct a KNN graph on features $\mathbf{F}$ using cosine distance: $\mathbf{A} = \texttt{KNN}_{\texttt{cos}}(\mathbf{F}, k)$, where $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{m+n}]$, $\mathbf{A}$ is the adjacency matrix of the resulting graph, k is the number of nearest neighbors, and $\texttt{KNN}_{\texttt{cos}}(\cdot)$ denotes the KNN algorithm with cosine distance. For simplicity, we consider a binary graph, *i.e.,* $\mathbf{A}_{ij} = 1$ if $\mathbf{A}_{ij} \in \texttt{topk}(\mathbf{A}_{i:})$; $\mathbf{A}_{ij} = 0$, otherwise.

3.2 Improved Text Embedding

The previous section addresses the challenge in graph generation, as outlined in Sec. 2.1. This section focuses on the challenge in text embedding, i.e., improving the finetuning of LMs.

Pseudo-Labeler. Leveraging the graphs generated in Sec. 3.1, we design a graph-based semi-supervised learning model to produce high-quality pseudo-labels, which enhance LM training. Given the node features $\mathbf{F}$, adjacency graph $\mathbf{A}$, and available labels $\mathcal{Y}_L = \{\mathbf{y}_i\}^m$, we train a GNN using the standard node classification loss:

$$\mathcal{L}_{\texttt{GNN}} = \sum_{i=1}^{m} \mathbf{y}_i^\top \ln \mathbf{z}_i, \quad \mathbf{z}_i = \text{softmax}(\texttt{GNN}_{\mathbf{\Phi}}(\mathbf{f}_i, \mathbf{A})), \tag{1}$$

where $\mathbf{\Phi}$ represents the GNN parameters. Here, any GNN model, such as graph convolutional networks (GCN) (Kipf & Welling, 2016), GraphSAGE (Hamilton et al., 2017), or graph attention networks (GAT) (Velickovic et al., 2017), can be employed (see Sec. 4.3 for experimental comparisons). Once trained, the pseudo labels $\hat{\mathbf{y}}_j$ for all unlabeled nodes ($j \in \{1, \ldots, n\}$) are generated by $\hat{\mathbf{y}}_j = \text{argmax}(\texttt{GNN}_{\mathbf{\Phi}}(\mathbf{f}_j, \mathbf{A}))$, enabling effective label propagation.

Text Embedding. To extract semantic-enriched text embeddings from documents, we finetune LMs on $\hat{\mathcal{D}} = \{\mathcal{X}_L, \mathcal{X}_U, \mathcal{Y}_L, \hat{\mathcal{Y}}_U\}$ where $\hat{\mathcal{Y}}_U = \{\hat{\mathbf{y}}_j\}^n$, using the following loss function:

$$\mathcal{L}_{\text{LM}} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{y}_i^\top \ln \mathbf{s}_i + \frac{1}{n}\sum_{j=1}^{n} \hat{\mathbf{y}}_j^\top \ln \mathbf{s}_j, \tag{2}$$

where $\mathbf{s}_i = \text{softmax}(\texttt{LM}_{\mathbf{\Theta}}(\mathbf{x}_i))$ and $\texttt{LM}_{\mathbf{\Theta}}(\cdot)$ is a pretrained LM with parameters $\mathbf{\Theta}$. After finetuning, for each node $\mathbf{x}_i$, $i \in \{1, 2, \ldots, m+n\}$, we generate its text embedding $\mathbf{e}_i$ by extracting the last hidden states of the finetuned LM, *i.e.,* $\mathbf{e}_i = \texttt{LM}_{\mathbf{\Theta}}(\mathbf{x}_i)$.

Following (He et al., 2024), we also utilize the large language model (LLM) GPT3.5 (Ouyang et al., 2022) to enhance text embeddings. We access GPT3.5 through its open API and treat it as an oracle, feeding it a prompt input $\mathbf{p}_i$ and obtaining the corresponding textual response $\mathbf{r}_i$ for each node i:

$$\mathbf{r}_i = \texttt{LLM}(\mathbf{p}_i), \quad \mathbf{p}_i = \text{concat}(\mathbf{x}_i, \mathbf{q}), \tag{3}$$

where the prompt $\mathbf{p}_i$ is concatenated by the text sequences $\mathbf{x}_i$ and a task-specific question $\mathbf{q}$ (He et al., 2024). The question $\mathbf{q}$ is standardized and universal for all nodes, which prompts GPT3.5 to classify each node and provide an explanation for its prediction (see Appendix for more details). Note that, we do not need to finetune GPT3.5 or access its inner structures or embeddings. Once the textual response $\mathbf{r}_i$ is obtained for $\mathbf{x}_i$, we replace $\mathbf{x}_i$ with $\mathbf{r}_i$ and finetune the $\texttt{LM}_{\mathbf{\Theta}}(\cdot)$ to generate updated text embeddings for the documents.

3.3 Classification

Up to this point, we have addressed the two key problems outlined in Sec. 2.1, obtaining semantic-enriched text embeddings $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{m+n}]$ and a meaningful graph $\mathbf{A}$. We can now tackle the classification task by training a graph-based semi-supervised learning model on $\mathbf{E}$ and $\mathbf{A}$ using the available labels $\mathcal{Y}_L$. Following (He et al., 2024), we utilize a GNN as the classification model. We also evaluate the performance of the LMs with generated pseudo-labels. Notably, the proposed SFGL can use an iterative training mechanism. The improved text embeddings facilitate the generation of higher-quality pseudo-labels, which, in turn, further enhance the discriminability of the embeddings. We explore this iterative training strategy in Sec. 4.3. Due to space limitations, the detailed training stages are summarized in Appendix A.5.

4 Experiments

4.1 Experimental Settings

Datasets. We conduct extensive experiments on four citation networks: `Cora` (Sen et al., 2008), `Pubmed` (Kipf & Welling, 2016), `ogbn-arxiv` (Hu et al., 2020), and `arxiv23` (He et al., 2024) (see Appendix A.1 for more details.). Following standard practice, we utilize traditional text encoders to process textual attributes for graph generation. For `Cora`, each publication is represented using a bag-of-words model (Harris, 1954) with a vocabulary of $1,433$ unique words. For `Pubmed`, documents are encoded as TF/IDF weighted word vectors (Aizawa, 2003) from a dictionary of 500 unique words. For `ogbn-arxiv` and `arxiv23`, textual features are extracted using a skip-gram model (Mikolov et al., 2013) and a word2vec model (Mikolov et al., 2013), respectively. Additional comparisons with other text encoders are provided in Sec. 4.3. In our experiments, the learning models have access to node attributes, while edge attributes remain unavailable.

Baselines. We compare our approach against several state-of-the-art methods. ① `GCN` (Kipf & Welling, 2016): trains a GCN using hand-crafted shallow features $\{\mathbf{f}_i\}^{m+n}$ and labels $\{\mathbf{y}_i\}^m$. ② `DeBERTa` (He et al., 2021): finetunes a pretrained `DeBERTa` on raw text attributes $\{\mathbf{x}_i\}^m$ and labels $\{\mathbf{y}_i\}^m$. ③ `GCN+DeBERTa` (Zhao et al., 2022): trains a GCN using text embeddings $\{\mathbf{e}_i\}^{m+n}$ and labels $\{\mathbf{y}_i\}^m$, where $\{\mathbf{e}_i\}^{m+n}$ are obtained by finetuning `DeBERTa` on $\{\mathbf{x}_i, \mathbf{y}_i\}^m$. ④ `DeBERTa+GPT` (Ouyang et al., 2022): finetunes a `DeBERTa` using `GPT`-generated responses $\{\mathbf{r}_i\}^m$ of labeled samples and labels $\{\mathbf{y}_i\}^m$. ⑤ `GCN+DeBERTa+GPT` (He et al., 2024): trains a GCN using text embeddings $\{\mathbf{e}_i\}^{m+n}$ and labels $\{\mathbf{y}_i\}^m$, where $\{\mathbf{e}_i\}^{m+n}$ are obtained by finetuning a `DeBERTa` on $\{\mathbf{r}_i, \mathbf{y}_i\}^m$. ⑥ `SFGL` (ours): trains a GCN using text embeddings $\{\mathbf{e}_i\}^{m+n}$ and labels $\{\mathbf{y}_i\}^m$, where $\{\mathbf{e}_i\}^{m+n}$ are obtained by finetuning a `DeBERTa` with pseudo-labels $\{\hat{\mathbf{y}}_j\}^n$. ⑦ `SFGL+GPT` (ours): similar to `SFGL` but uses `GPT`-generated responses to finetune the `DeBERTa`.

Table 1: Performance of different models trained on four datasets with varying labeling rates, where "— — —" indicates unreliable results due to an extremely limited number of training labels.

Datasets & Method	Labeling Rates (labels)				
`Cora`	2.58%$_{(70)}$	5.17%$_{(140)}$	7.75%$_{(210)}$	15.51%$_{(420)}$	22.16%$_{(600)}$
`GCN`	0.5898$_{\pm 0.0159}$	0.6038$_{\pm 0.0116}$	0.6358$_{\pm 0.0164}$	0.6936$_{\pm 0.0055}$	0.7114$_{\pm 0.0049}$
`DeBERTa`	0.2406$_{\pm 0.0799}$	0.3048$_{\pm 0.0198}$	0.3668$_{\pm 0.0632}$	0.6016$_{\pm 0.0495}$	0.6904$_{\pm 0.0413}$
`GCN+DeBERTa`	0.2810$_{\pm 0.0136}$	0.3598$_{\pm 0.0655}$	0.4744$_{\pm 0.1281}$	0.6482$_{\pm 0.0268}$	0.6940$_{\pm 0.0286}$
`SFGL(ours)`	0.6086$_{\pm 0.0140}$	0.6314$_{\pm 0.0191}$	0.6612$_{\pm 0.0206}$	0.6942$_{\pm 0.0101}$	0.7204$_{\pm 0.0153}$
`DeBERTa+GPT`	0.2776$_{\pm 0.0128}$	0.5450$_{\pm 0.1121}$	0.6624$_{\pm 0.0067}$	0.6888$_{\pm 0.0189}$	0.7082$_{\pm 0.0184}$
`GCN+DeBERTa+GPT`	0.4862$_{\pm 0.0489}$	0.6272$_{\pm 0.0217}$	0.6550$_{\pm 0.0077}$	0.6836$_{\pm 0.0193}$	0.6978$_{\pm 0.0085}$
`SFGL+GPT(ours)`	0.6348$_{\pm 0.0136}$	0.6856$_{\pm 0.0131}$	0.6992$_{\pm 0.0265}$	0.7278$_{\pm 0.0041}$	0.7374$_{\pm 0.0072}$
`Pubmed`	0.30%$_{(60)}$	0.61%$_{(120)}$	0.91%$_{(180)}$	1.83%$_{(360)}$	2.54%$_{(500)}$
`GCN`	0.6770$_{\pm 0.0217}$	0.7022$_{\pm 0.0107}$	0.7398$_{\pm 0.0096}$	0.7388$_{\pm 0.0041}$	0.7464$_{\pm 0.0100}$
`DeBERTa`	0.3616$_{\pm 0.0976}$	0.4114$_{\pm 0.0532}$	0.4402$_{\pm 0.0571}$	0.5732$_{\pm 0.0363}$	0.7474$_{\pm 0.0832}$
`GCN+DeBERTa`	0.4702$_{\pm 0.0112}$	0.5046$_{\pm 0.0180}$	0.5520$_{\pm 0.0477}$	0.6648$_{\pm 0.0453}$	0.8042$_{\pm 0.0737}$
`SFGL(ours)`	0.7238$_{\pm 0.0124}$	0.7792$_{\pm 0.0263}$	0.8188$_{\pm 0.0190}$	0.8524$_{\pm 0.0280}$	0.8390$_{\pm 0.0204}$
`DeBERTa+GPT`	0.4514$_{\pm 0.0700}$	0.5206$_{\pm 0.1020}$	0.6196$_{\pm 0.1403}$	0.9278$_{\pm 0.0034}$	0.9274$_{\pm 0.0032}$
`GCN+DeBERTa+GPT`	0.7386$_{\pm 0.0129}$	0.7440$_{\pm 0.0272}$	0.7868$_{\pm 0.0212}$	0.9242$_{\pm 0.0049}$	0.9282$_{\pm 0.0022}$
`SFGL+GPT(ours)`	0.8640$_{\pm 0.0381}$	0.9014$_{\pm 0.0064}$	0.9088$_{\pm 0.0037}$	0.9162$_{\pm 0.0059}$	0.9116$_{\pm 0.0027}$
`ogbn-arxiv`	0.03%$_{(50)}$	0.06%$_{(100)}$	0.09%$_{(150)}$	0.18%$_{(300)}$	0.30%$_{(500)}$
`GCN`	0.3375$_{\pm 0.0360}$	0.3578$_{\pm 0.0179}$	0.3991$_{\pm 0.0214}$	0.4227$_{\pm 0.0132}$	0.4516$_{\pm 0.0087}$
`DeBERTa`	— — —	— — —	— — —	— — —	0.2541$_{\pm 0.0636}$
`GCN+DeBERTa`	0.1661$_{\pm 0.0584}$	0.1903$_{\pm 0.0523}$	0.1964$_{\pm 0.0464}$	0.2130$_{\pm 0.0169}$	0.2359$_{\pm 0.0240}$
`SFGL(ours)`	0.3251$_{\pm 0.0373}$	0.4171$_{\pm 0.0275}$	0.4626$_{\pm 0.0379}$	0.4979$_{\pm 0.0238}$	0.5577$_{\pm 0.0060}$
`DeBERTa+GPT`	— — —	— — —	— — —	0.1653$_{\pm 0.0575}$	0.2892$_{\pm 0.0182}$
`GCN+DeBERTa+GPT`	0.2137$_{\pm 0.0317}$	0.2200$_{\pm 0.0449}$	0.2188$_{\pm 0.0709}$	0.2321$_{\pm 0.0929}$	0.2644$_{\pm 0.0185}$
`SFGL+GPT(ours)`	0.4570$_{\pm 0.0625}$	0.6007$_{\pm 0.0172}$	0.6380$_{\pm 0.0159}$	0.6561$_{\pm 0.0145}$	0.6567$_{\pm 0.0182}$
`arxiv23`	0.11%$_{(50)}$	0.22%$_{(100)}$	0.32%$_{(150)}$	0.65%$_{(300)}$	1.08%$_{(500)}$
`GCN`	0.3809$_{\pm 0.0251}$	0.4229$_{\pm 0.0137}$	0.4342$_{\pm 0.0092}$	0.4771$_{\pm 0.0076}$	0.4976$_{\pm 0.0093}$
`DeBERTa`	— — —	— — —	0.2204$_{\pm 0.0148}$	0.2972$_{\pm 0.0339}$	0.5619$_{\pm 0.0198}$
`GCN+DeBERTa`	0.2506$_{\pm 0.0146}$	0.2659$_{\pm 0.0102}$	0.2648$_{\pm 0.0067}$	0.2803$_{\pm 0.0040}$	0.2976$_{\pm 0.0163}$
`SFGL(ours)`	0.4323$_{\pm 0.0259}$	0.5002$_{\pm 0.0125}$	0.5160$_{\pm 0.0127}$	0.5699$_{\pm 0.0036}$	0.5973$_{\pm 0.0080}$
`DeBERTa+GPT`	— — —	— — —	0.2223$_{\pm 0.0156}$	0.3967$_{\pm 0.0417}$	0.6137$_{\pm 0.0497}$
`GCN+DeBERTa+GPT`	0.3261$_{\pm 0.0389}$	0.3474$_{\pm 0.0404}$	0.3851$_{\pm 0.0123}$	0.3974$_{\pm 0.0307}$	0.4686$_{\pm 0.0199}$
`SFGL+GPT(ours)`	0.4953$_{\pm 0.0066}$	0.5811$_{\pm 0.0167}$	0.6041$_{\pm 0.0164}$	0.6439$_{\pm 0.0133}$	0.6552$_{\pm 0.0247}$

4.2 Comparison Analysis

Table 1 presents the classification performance of various methods across different datasets and labeling rates. Our key observations are as follows: ① Across all datasets, classification accuracy generally improves as the labeling rate increases. This aligns with expectations, as more labeled data provides better supervision, enhancing model performance. ② The baselines `DeBERTa` and `DeBERTa+GPT` perform poorly at low labeling rates, particularly on `ogbn-arxiv` and `arxiv23`, where finetuning LMs becomes unreliable under extreme label scarcity. This is because insufficient labeled data fails to provide sufficient information for gradient updates, leading to the learning of overly narrow patterns, which impairs generalization. However, incorporating additional pseudo-labels enables LMs to generate informative text embeddings, thereby enhancing performance. ③ Incorporating GPT-generated responses significantly enhances both `GCN` and `DeBERTa`, particularly on the `Pubmed` dataset. This highlights the potential of LLMs to extract and leverage rich semantic information for text modeling. ④ In most cases, the proposed `SFGL` and `SFGL+GPT` achieve the highest performance, demonstrating their effectiveness in low-supervision scenarios.

4.3 Ablation Study

Iterative Performance. As mentioned in Sec. 3.3, improving the GNNs allows us to generate more accurate pseudo-labels. These enhanced pseudo-labels, in turn, enable us to retrain the LMs, pro-

Table 2: Performance comparison of the proposed method with and without iterative training.

Labeling Rates (labels)	0.30%$_{(60)}$	0.61%$_{(120)}$	0.91%$_{(180)}$	1.83%$_{(360)}$	2.54%$_{(500)}$
SFGL	$0.7238_{\pm 0.0124}$	$0.7792_{\pm 0.0263}$	$0.8188_{\pm 0.0190}$	$0.8524_{\pm 0.0280}$	$0.8390_{\pm 0.0204}$
SFGL+$\underline{\text{GCN}}$	$0.7434_{\pm 0.0048}$	$0.8510_{\pm 0.0096}$	$0.8686_{\pm 0.0263}$	$0.8976_{\pm 0.0084}$	$0.8966_{\pm 0.0070}$
SFGL+$\underset{\sim}{\text{GCN}}$	$0.8912_{\pm 0.0293}$	$0.8974_{\pm 0.0116}$	$0.9110_{\pm 0.0083}$	$0.9066_{\pm 0.0094}$	$0.9186_{\pm 0.0073}$
SFGL+GPT	$0.8640_{\pm 0.0381}$	$0.9014_{\pm 0.0064}$	$0.9088_{\pm 0.0037}$	$0.9162_{\pm 0.0059}$	$0.9116_{\pm 0.0027}$
SFGL+GPT+$\underline{\text{GCN}}$	$0.9142_{\pm 0.0086}$	$0.9250_{\pm 0.0043}$	$0.9294_{\pm 0.0040}$	$0.9290_{\pm 0.0032}$	$0.9374_{\pm 0.0015}$
SFGL+GPT+$\underset{\sim}{\text{GCN}}$	$0.9138_{\pm 0.0029}$	$0.9198_{\pm 0.0054}$	$0.9216_{\pm 0.0080}$	$0.9240_{\pm 0.0046}$	$0.9264_{\pm 0.0026}$

Table 3: Performance comparison of different models using various distance metrics, including Manhattan (M), Euclidean (E), and Cosine (C) distances.

Labeling Rates (labels)	2.58%$_{(70)}$	5.17%$_{(140)}$	7.75%$_{(210)}$	15.51%$_{(420)}$	22.16%$_{(600)}$
SFGL(M)	$0.5454_{\pm 0.0327}$	$0.5528_{\pm 0.1459}$	$0.6206_{\pm 0.0101}$	$0.6810_{\pm 0.0080}$	$0.7144_{\pm 0.0109}$
SFGL(E)	$0.5178_{\pm 0.0271}$	$0.5524_{\pm 0.1468}$	$0.6240_{\pm 0.0163}$	$0.6718_{\pm 0.0136}$	$0.7138_{\pm 0.0141}$
SFGL(C)	$0.6086_{\pm 0.0140}$	$0.6314_{\pm 0.0191}$	$0.6612_{\pm 0.0206}$	$0.6942_{\pm 0.0101}$	$0.7204_{\pm 0.0153}$
SFGL+GPT(M)	$0.6176_{\pm 0.0291}$	$0.6564_{\pm 0.0223}$	$0.6666_{\pm 0.0229}$	$0.6954_{\pm 0.0215}$	$0.7214_{\pm 0.0144}$
SFGL+GPT(E)	$0.6138_{\pm 0.0168}$	$0.6474_{\pm 0.0132}$	$0.6588_{\pm 0.0110}$	$0.6932_{\pm 0.0249}$	$0.7238_{\pm 0.0098}$
SFGL+GPT(C)	$0.6348_{\pm 0.0136}$	$0.6856_{\pm 0.0131}$	$0.6992_{\pm 0.0265}$	$0.7278_{\pm 0.0041}$	$0.7374_{\pm 0.0072}$
DeBERTa+GCN(M)	$0.4956_{\pm 0.0136}$	$0.5286_{\pm 0.1312}$	$0.6134_{\pm 0.0101}$	$0.6922_{\pm 0.0101}$	$0.7282_{\pm 0.0080}$
DeBERTa+GCN(E)	$0.4660_{\pm 0.0261}$	$0.5126_{\pm 0.0182}$	$0.6194_{\pm 0.0109}$	$0.6882_{\pm 0.0087}$	$0.7254_{\pm 0.0074}$
DeBERTa+GCN(C)	$0.6232_{\pm 0.0094}$	$0.6616_{\pm 0.0077}$	$0.6728_{\pm 0.0079}$	$0.7064_{\pm 0.0077}$	$0.7214_{\pm 0.0096}$
DeBERTa+GPT+GCN(M)	$0.5380_{\pm 0.0135}$	$0.6480_{\pm 0.0034}$	$0.6656_{\pm 0.0122}$	$0.7092_{\pm 0.0088}$	$0.7332_{\pm 0.0109}$
DeBERTa+GPT+GCN(E)	$0.5268_{\pm 0.0146}$	$0.6412_{\pm 0.0110}$	$0.6634_{\pm 0.0077}$	$0.7112_{\pm 0.0131}$	$0.7318_{\pm 0.0110}$
DeBERTa+GPT+GCN(C)	$0.6588_{\pm 0.0113}$	$0.6978_{\pm 0.0200}$	$0.6962_{\pm 0.0079}$	$0.7206_{\pm 0.0109}$	$0.7426_{\pm 0.0072}$

Table 4: Performance comparison between real graphs and KNN graphs.

Labeling Rates (labels)	0.11%$_{(50)}$	0.22%$_{(100)}$	0.32%$_{(150)}$	0.65%$_{(300)}$	1.08%$_{(500)}$
Real Graph	$0.3691_{\pm 0.0033}$	$0.4124_{\pm 0.0185}$	$0.4354_{\pm 0.0107}$	$0.4764_{\pm 0.0116}$	$0.5039_{\pm 0.0071}$
KNN Graph k=10	$0.3660_{\pm 0.0136}$	$0.4031_{\pm 0.0054}$	$0.4152_{\pm 0.0145}$	$0.4541_{\pm 0.0114}$	$0.4706_{\pm 0.0164}$
KNN Graph k=20	$0.3809_{\pm 0.0251}$	$0.4229_{\pm 0.0137}$	$0.4342_{\pm 0.0092}$	$0.4771_{\pm 0.0076}$	$0.4976_{\pm 0.0093}$
KNN Graph k=30	$0.3795_{\pm 0.0278}$	$0.4085_{\pm 0.0026}$	$0.4130_{\pm 0.0099}$	$0.4595_{\pm 0.0149}$	$0.4864_{\pm 0.0097}$
KNN Graph k=50	$0.3818_{\pm 0.0149}$	$0.4081_{\pm 0.0081}$	$0.4203_{\pm 0.0133}$	$0.4679_{\pm 0.0194}$	$0.4857_{\pm 0.0207}$
KNN Graph k=100	$0.3767_{\pm 0.0182}$	$0.4100_{\pm 0.0055}$	$0.4201_{\pm 0.0091}$	$0.4778_{\pm 0.0097}$	$0.5005_{\pm 0.0119}$

ducing more refined text embeddings (see Appendix A.5 for detailed training stages). To evaluate this iterative training strategy, we introduce two improved GNN variants, $\underline{\text{GCN}}$ and $\underset{\sim}{\text{GCN}}$, derived from SFGL and SFGL+GPT, respectively, as new pseudo-labelers. Table 2 presents the iterative performance of our models on the Pubmed dataset. Our results demonstrate that this iterative mechanism brings further performance improvements, highlighting the mutually beneficial relationship between GNNs and LMs in iterative training. Notably, SFGL+GPT+$\underline{\text{GCN}}$ consistently outperforms SFGL+GPT+$\underset{\sim}{\text{GCN}}$, indicating slight overfitting during the iterative training process.

Different Distance Metrics. As discussed in Sec. 2.3, various distance metrics, such as Euclidean, Manhattan, and Cosine distances, can be used in KNN graphs to measure similarity between samples. To evaluate their impact, we compare model performance using these different metrics. Table 3 presents the results for SFGL, SFGL+GPT, DeBERTa+GCN, and DeBERTa+GPT+GCN. The first two models use GCNs as the final classifier, while the latter two employ LMs as the classifier but leverage our graph-based pseudo-labeler to enhance LM finetuning. As shown, Cosine distance consistently outperforms the other metrics, particularly at very low labeling rates, further justifying its effectiveness for graph construction in our framework.

Real Graph vs. KNN Graph. Table 4 presents the performance comparison between real graphs and KNN graphs on the arxiv23 dataset, considering different labeling rates and values of k. Notably, under extremely low labeling rates, the performance achieved with the true graph is not significantly better—and in some cases, even worse—than that obtained with a pre-constructed KNN graph. This suggests that a true graph may not be crucial for improving LM finetuning, particularly in scenarios with very limited supervision. Our explanation for this phenomenon is that both the number of labeled nodes and the choice of which nodes to label play a crucial role in determining the final classification performance. Consider an extreme case: if all labeled nodes in a real graph are isolated, label information cannot propagate to the unlabeled nodes. In contrast, constructing an artificial graph for these nodes may establish connections between the unlabeled nodes and the isolated labeled nodes, facilitating label propagation across the graph. Consequently, artificial graphs can potentially outperform real graphs, particularly in scenarios with very limited supervision. See Appendix A.4 for additional experimental results.

5 Related Work

Latent Graph Inference Models. Traditional LGI methods adopt linear projections to learn node representations and optimize various objective functions to learn latent graphs. For example, Zhang et al. (2010) design an entropy regularization to control the uniformity level of edge weights. Nie et al. (2016) infer an optimal graph by assigning adaptive neighbors. Lu et al. (2018) impose spectral sparsity on the graph Laplacian matrix. Lu et al. (2021a) assume that two samples are likely to belong to the same class if one sample is close to the reconstructed representation of the other. Advanced LGI models exploit GNNs to learn the latent graphs. For instance, Franceschi et al. (2019) model a discrete probability distribution for the edges. Fatemi et al. (2021) provide supplementary supervision for latent graphs through a self-supervision task. Lu et al. (2023) propose a model-agnostic model that obtains supplementary supervision directly from true labels. However, existing methods typically rely on artificial assumptions about the underlying edge distribution. For example, Zhang et al. (2010) impose a uniformity assumption on edge weights. Lu et al. (2018) introduce a block diagonal prior to the graph Laplacian matrix. Lu et al. (2021b) construct an adaptive neighborhood graph by assuming the probability property of edge weights. Fatemi et al. (2021) assume that a graph structure effective for predicting features is also effective for label prediction. These artificial assumptions may not accurately reflect real graph structures and require specific optimizations with additional model training across the entire dataset.

Language-Assisted Graph Models. LGI models typically rely on feature engineering approaches, such as skip-gram and TF-IDF, to encode textual sequences into feature vectors. In contrast, recent LAG models seek to enhance text embeddings by leveraging various LMs to extract richer semantic features from text sequences (Li et al., 2024). For example, Zhao et al. (2022) design a variational expectation-maximization framework to fuse graph structure and language learning for classification. Duan et al. (2023) first conduct parameter-efficient finetuning on a pretrained LM and then generate text embeddings using the finetuned LM. He et al. (2024) leverages an LLM to capture textual information and applies a small LM as the interpreter to transform the responses of the LLM into informative features. Yu et al. (2025) use LLMs to generate nodes and edges for text-attributed graphs, which harnesses LLMs for enhancing class-level information. For most existing LAG models, finetuning an LM on the target dataset is essential to generate semantically enriched text embeddings. However, it is notorious that finetuning a pretrained LM typically demands a large amount of annotated data (Brown et al., 2020), which poses a significant challenge for LAG models in semi-supervised learning tasks, where available annotated data is often scarce.

6 Conclusion

In this paper, we identify two primary challenges in exiting graph-language models for semi-supervised learning, i.e., artificial structural assumptions in graph generation and unreliable LM finetuning for text embedding. We tackle these challenges by establishing a well-grounded structural prior. Specifically, we examine the scale-free property of citation networks and reveal that this structural characteristic can be effectively approximated using a simple KNN graph. Building on this observation, we propose a novel graph-language model that employs a customized KNN algorithm for scale-free graph generation and utilizes a graph-based pseudo-labeler to provide additional supervision for improved text embedding. Extensive experiments on representative datasets validate our findings on the scale-free structural approximation of KNN graphs and demonstrate the effectiveness of integrating graph generation and text embedding with a real structural prior. We hope this study highlights the synergistic potential of GNNs and LMs, providing valuable insights for researchers in both the GNN and NLP communities.

REFERENCES

Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65, 2003.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999.

Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific American*, 288(5):60–69, 2003.

Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge University Press, Cambridge, 2016. ISBN 9781107076266 1107076269.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

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

Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 22667–22681, 2021.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1972–1982, 2019.

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

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

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *International Conference on Learning Representations*, 2024.

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

Norman P Hummon and Patrick Dereian. Connectivity in a citation network: The development of dna theory. *Social networks*, 11(1):39–63, 1989.

Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M. Bronstein. Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1606–1617, 2023.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

Chaofan Li, Minghao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. Making text embedders few-shot learners. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. A survey of graph meets large language model: Progress and future directions, 2024. URL https://arxiv.org/abs/2311.12399.

Canyi Lu, Jiashi Feng, Zhouchen Lin, Tao Mei, and Shuicheng Yan. Subspace clustering by block diagonal representation. *IEEE transactions on pattern analysis and machine intelligence*, 41(2): 487–501, 2018.

Jianglin Lu, Jingxu Lin, Zhihui Lai, Hailing Wang, and Jie Zhou. Target redirected regression with dynamic neighborhood structure. *Information Sciences*, 544:564–584, 2021a.

Jianglin Lu, Hailing Wang, Jie Zhou, Yudong Chen, Zhihui Lai, and Qinghua Hu. Low-rank adaptive graph embedding for unsupervised feature extraction. *Pattern Recognition*, 113:107758, 2021b. ISSN 0031-3203.

Jianglin Lu, Yi Xu, Huan Wang, Yue Bai, and Yun Fu. Latent graph inference with limited supervision. In *Advances in Neural Information Processing Systems*, 2023.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Feiping Nie, Wei Zhu, and Xuelong Li. Unsupervised feature selection with structured graph optimization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Yijian Qin, Xin Wang, Ziwei Zhang, and Wenwu Zhu. Disentangled representation learning with large language models for text-attributed graphs. *arXiv*, abs/2310.18152, 2023.

Filippo Radicchi, Santo Fortunato, and Alessandro Vespignani. Citation networks. *Models of science dynamics: Encounters between complexity theory and information sciences*, pp. 233–257, 2011.

S. Redner. How popular is your paper? an empirical study of the citation distribution. *The European Physical Journal B*, 4(2):131–134, August 1998. ISSN 1434-6028.

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

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11897–11916, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

Jianxiang Yu, Yuxiang Ren, Chenghua Gong, Jiaqi Tan, Xiang Li, and Xuecang Zhang. Leveraging large language models for node generation in few-shot learning on text-attributed graphs. In *Proceedings of the AAAI conference on artificial intelligence*, 2025.

Limei Zhang, Lishan Qiao, and Songcan Chen. Graph-optimized locality preserving projections. *Pattern Recognition*, 43(6):1993–2002, 2010.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. In *The Eleventh International Conference on Learning Representations*, 2022.

A APPENDIX

A.1 DATASET DESCRIPTION

We conducted extensive experiments on four text-attributed graph benchmarks: `Cora` (Sen et al., 2008), `PubMed` (Kipf & Welling, 2016), `ogbn-arxiv` (Hu et al., 2020), and `arxiv23` (He et al., 2024). The `Cora` dataset contains $2,708$ papers belonging to seven different classes. The `PubMed` dataset includes $19,717$ publications categorized into three classes. The `ogbn-arxiv` (Hu et al., 2020) dataset includes $169,343$ arXiv papers across 40 subject areas in computer science. Labels for this dataset are manually annotated by the authors and arXiv. The `arxiv23` (He et al., 2024) dataset includes all computer science papers submitted in 2023 or later on arXiv. Similar to ogbn-arxiv, these papers are divided into 40 classes.

A.2 MODEL CONFIGURATION

We follow (He et al., 2024) for model configurations. For GNNs, we use a two-layer GCN (see Sec. 4.3 for comparisons with other GNN architectures). The hyper-parameters for hidden dimension, learning rate, dropout ratio, and weight decay are set to 128, 0.001, 0.5, and 0.0005, respectively. For LMs, we finetune a pretrained DeBERTa (He et al., 2021) on the target datasets. The batch size, learning rate, and dropout ratio are set to 20, 2×10^{-5}, and 0.3, respectively. We also employ GPT3.5 (Ouyang et al., 2022) for inference. We empirically set k to $25, 15, 25, 20$ for `Cora`, `Pubmed`, `ogbn-arxiv`, and `arxiv23`, respectively. We conduct five independent experiments with different random seeds and report the average test accuracy along with the standard deviation.

For LLM prompt inputs, we follow established methods (He et al., 2024). Each prompt consists of a paper's title and abstract, along with a task-specific question designed to elicit a class prediction and a corresponding explanation from the LLM. The generated textual responses by the LLMs serve as inputs for the LMs, which use them to generate text embeddings for all nodes. These embeddings are then further processed by GNNs to enhance model performance.

A.3 PROOF OF PROPOSITION

Proposition 1. *For a scale-free network, assuming that there is at most one node v whose degree belongs to $[\theta_{max}, \infty)$, we have $\theta_{max} = \theta_{min}|\mathcal{V}|^{\frac{1}{\alpha-1}}$, where θ_{max} and θ_{min} refer to the maximum and minimal degree in the scale-free network.*

Proof. We write the degree distribution $\mathbb{P}(\theta)$ of a scale free network as: $\mathbb{P}(\theta) = \xi\theta^{-\alpha}$, where θ is the degree of a node, ξ is a constant, and α is a scaling parameter of the distribution. Since $\int \theta^{-\alpha} d\theta = \frac{\theta^{1-\alpha}}{1-\alpha} + constant$ and $\int_{\theta_{min}}^{\infty} \mathbb{P}(\theta) d\theta = 1$, we have $\frac{1}{\xi} = \int_{\theta_{min}}^{\infty} \theta^{-\alpha} d\theta = [\frac{\theta^{1-\alpha}}{1-\alpha}]_{\theta_{min}}^{\infty}$. Since $\lim_{\theta\to\infty} \frac{\theta^{1-\alpha}}{1-\alpha} = 0 \ (\alpha > 1)$, we have $[\frac{\theta^{1-\alpha}}{1-\alpha}]_{\theta_{min}}^{\infty} = \frac{\theta_{min}^{1-\alpha}}{\alpha-1}$. As a result, we obtain $\xi = (\alpha-1)\theta_{min}^{\alpha-1}$ and $\mathbb{P}(\theta) = (\alpha-1)\theta_{min}^{\alpha-1}\theta^{-\alpha}$. Assume that there is at most one node whose degree belongs to $[\theta_{max}, \infty)$, where θ_{max} is the maximum degree in the network. We have $\int_{\theta_{max}}^{\infty} \mathbb{P}(\theta) d\theta = \int_{\theta_{max}}^{\infty} \xi\theta^{-\alpha} d\theta = \xi\frac{\theta_{max}^{1-\alpha}}{\alpha-1} = (\alpha-1)\theta_{min}^{\alpha-1}\frac{\theta_{max}^{1-\alpha}}{\alpha-1} = \frac{1}{|\mathcal{V}|}$, where $|\mathcal{V}|$ is the number of nodes. After simplification, we obtain $(\frac{\theta_{min}}{\theta_{max}})^{\alpha-1} = \frac{1}{|\mathcal{V}|}$, which leads to $\theta_{max} = \theta_{min}|\mathcal{V}|^{\frac{1}{\alpha-1}}$. □

Proposition 2. *For a KNN graph, we have $\sum_{v\in\mathcal{V}} C_{ID}(v) = \sum_{v\in\mathcal{V}} C_D(v) - \sum_{v\in\mathcal{V}} C_{OD}(v) = 2|\mathcal{E}| - k|\mathcal{V}|$, where $|\mathcal{E}| \leq k|\mathcal{V}|$.*

Proof. In a KNN graph, we have $C_D(v) = C_{ID}(v) + C_{OD}(v)$, and $\sum_{v\in\mathcal{V}} C_D(v) = 2|\mathcal{E}|$. For $\forall v \in \mathcal{V}$, $C_{OD}(v) = k$. Considering that two nodes may regard each other as its nearest neighbors, the total number of edges is at most $k|\mathcal{V}|$, i.e., $|\mathcal{E}| \leq k|\mathcal{V}|$. Based on the above points, we can derive: $\sum_{v\in\mathcal{V}} C_{ID}(v) = \sum_{v\in\mathcal{V}} C_D(v) - \sum_{v\in\mathcal{V}} C_{OD}(v) = 2|\mathcal{E}| - k|\mathcal{V}|$, where $|\mathcal{E}| \leq k|\mathcal{V}|$. □

Table 5: Performance comparison of different models using different types of GNNs.

Labeling Rates (labels)	2.58%$_{(70)}$	5.17%$_{(140)}$	7.75%$_{(210)}$	15.51%$_{(420)}$	22.16%$_{(600)}$
GCN	0.5898$_{\pm 0.0159}$	0.6038$_{\pm 0.0116}$	0.6358$_{\pm 0.0164}$	0.6936$_{\pm 0.0055}$	0.7114$_{\pm 0.0049}$
SAGE	0.5660$_{\pm 0.0109}$	0.5994$_{\pm 0.0143}$	0.6326$_{\pm 0.0150}$	0.6910$_{\pm 0.0106}$	0.7044$_{\pm 0.0153}$
GAT	0.6050$_{\pm 0.0052}$	0.6262$_{\pm 0.0074}$	0.6560$_{\pm 0.0075}$	0.6852$_{\pm 0.0092}$	0.6944$_{\pm 0.0174}$
SFGL(GCN)	0.6086$_{\pm 0.0140}$	0.6314$_{\pm 0.0191}$	0.6612$_{\pm 0.0206}$	0.6942$_{\pm 0.0101}$	0.7204$_{\pm 0.0153}$
SFGL(SAGE)	0.5794$_{\pm 0.0103}$	0.6502$_{\pm 0.0100}$	0.6706$_{\pm 0.0072}$	0.7056$_{\pm 0.0135}$	0.7234$_{\pm 0.0150}$
SFGL(GAT)	0.5974$_{\pm 0.0117}$	0.6586$_{\pm 0.0099}$	0.6998$_{\pm 0.0079}$	0.7148$_{\pm 0.0200}$	0.7196$_{\pm 0.0081}$
SFGL+GPT(GCN)	0.6348$_{\pm 0.0136}$	0.6856$_{\pm 0.0131}$	0.6992$_{\pm 0.0265}$	0.7278$_{\pm 0.0041}$	0.7374$_{\pm 0.0072}$
SFGL+GPT(SAGE)	0.6342$_{\pm 0.0153}$	0.6806$_{\pm 0.0142}$	0.7050$_{\pm 0.0080}$	0.7276$_{\pm 0.0162}$	0.7348$_{\pm 0.0110}$
SFGL+GPT(GAT)	0.6390$_{\pm 0.0223}$	0.6954$_{\pm 0.0136}$	0.7128$_{\pm 0.0199}$	0.7256$_{\pm 0.0114}$	0.7390$_{\pm 0.0076}$
DeBERTa+GCN	0.6232$_{\pm 0.0094}$	0.6616$_{\pm 0.0077}$	0.6728$_{\pm 0.0079}$	0.7064$_{\pm 0.0077}$	0.7214$_{\pm 0.0096}$
DeBERTa+SAGE	0.5882$_{\pm 0.0081}$	0.6396$_{\pm 0.0100}$	0.6652$_{\pm 0.0090}$	0.7122$_{\pm 0.0074}$	0.7224$_{\pm 0.0091}$
DeBERTa+GAT	0.6112$_{\pm 0.0057}$	0.6530$_{\pm 0.0104}$	0.6986$_{\pm 0.0128}$	0.7074$_{\pm 0.0132}$	0.7186$_{\pm 0.0065}$
DeBERTa+GPT+GCN	0.6588$_{\pm 0.0113}$	0.6978$_{\pm 0.0200}$	0.6962$_{\pm 0.0079}$	0.7206$_{\pm 0.0109}$	0.7426$_{\pm 0.0072}$
DeBERTa+GPT+SAGE	0.6500$_{\pm 0.0113}$	0.6918$_{\pm 0.0051}$	0.6914$_{\pm 0.0122}$	0.7166$_{\pm 0.0063}$	0.7312$_{\pm 0.0060}$
DeBERTa+GPT+GAT	0.6482$_{\pm 0.0103}$	0.6910$_{\pm 0.0033}$	0.7114$_{\pm 0.0066}$	0.7118$_{\pm 0.0059}$	0.7294$_{\pm 0.0051}$

A.4 Additional Experimental Results

As mentioned in Sec. 3.2, various GNNs can be used to implement the pseudo-labeler. To assess their impact, we evaluate model performance using different GNNs, including GCN, SAGE, and GAT. Table 5 presents the comparison results across these variants. Our findings indicate that the proposed models exhibit robustness to the choice of GNN, as they achieve similar performance improvements regardless of the specific architecture used.

In Sec. 4.1, our experiments employ traditional text encoders for graph construction and pseudo-labeling. In fact, a variety of text encoders can be utilized for this task. To evaluate their impact, we experiment with several text encoders, including a traditional TF-IDF encoder (Enc1), a pretrained DeBERTa (Enc2), a pretrained E5-Mistral-7B-Instruct (Enc3) (Wang et al., 2024), a pretrained bge-en-icl (Enc4) (Li et al., 2025), and a fine-tuned DeBERTa (Enc5). Table 6 presents the comparison results on the PubMed dataset. As observed, pretrained models such as DeBERTa, E5-Mistral-7B-Instruct, and bge-en-icl fail to achieve satisfactory results. In contrast, the fine-tuned DeBERTa model demonstrates significant improvement, with TF-IDF performance falling between that of the pretrained and finetuned models. However, finetuning DeBERTa requires a sufficient number of labels, which is why we initially use shallow text encoders to facilitate pseudo-label generation. Notably, the two variants of our proposed SFGL listed in Table 2 use the finetuned DeBERTa as their text encoder. As shown, incorporating a more refined text encoder leads to further performance gains. Additionally, both pretrained E5-Mistral-7B-Instruct and bge-en-icl outperform the pretrained DeBERTa model but fall short of the finetuned DeBERTa. It is reasonable to infer that a finetuned E5-Mistral-7B-Instruct or bge-en-icl model would surpass the finetuned DeBERTa in performance.

Table 6: Results across different text encoders.

Labels	60	120	180	360	500
Enc1	0.665	0.696	0.748	0.749	0.758
Enc2	0.475	0.518	0.534	0.538	0.548
Enc3	0.550	0.614	0.648	0.686	0.690
Enc4	0.596	0.636	0.648	0.652	0.667
Enc5	0.743	0.815	0.856	0.877	0.876

Figures 6 and 7 illustrate the degree distribution and fitting curves for $k = 20$. As shown, the degree distributions of KNN graphs with a larger k still approximate a scale-free structure. We further analyze the KNN graph on the Cora dataset using Euclidean distance as the similarity metric. As depicted in Fig. 8 (b), the degree distribution of the KNN graph ($k = 5$) constructed with Euclidean distance does not follow a power law pattern. Notably, $2,080$ nodes have zero in-degrees, meaning they are not among the top 5 nearest neighbors of any nodes. This phenomenon likely arises from the way text embeddings encode semantic and syntactic information, which simple Euclidean distance fails to capture effectively. In other words, raw Euclidean distances between embedding vectors do not adequately reflect semantic similarities. This is because Euclidean distance primarily measures

Algorithm 1 Scale-Free Graph-Language Model (SFGL / SFGL+GPT)

```
Require: Input sequences $\mathbf{X}$, labels $\mathbf{Y}_L$, the number of neighbors $k$, query question $\mathbf{q}$, prompt $\mathbf{p}_i$.
 1: Encode the text sequences of nodes into hand-crafted features: $\mathbf{F} = \mathtt{Enc}(\mathbf{X})$.
 2: Construct a KNN graph with Cosine distance metric: $\mathbf{A} = \mathtt{KNN}_{\mathtt{cos}}(\mathbf{F}, k)$.
 3: Train a GNN ($\mathtt{GNN}_{\mathbf{\Phi}}$) on $\mathbf{F}$, $\mathbf{A}$, and $\mathbf{Y}_L$ with loss function in Eq. 1.
 4: Generate pseudo labels for the unlabeled nodes: $\hat{\mathbf{y}}_j = \mathrm{argmax}(\mathtt{GNN}_{\mathbf{\Phi}}(\mathbf{f}_j, \mathbf{A})), j \in \{1, \cdots, n\}$.
 5: (Optional) Generate textual responses from an LLM : $\mathbf{r}_i = \mathtt{LLM}(\mathbf{p}_i),\ \mathbf{p}_i = \mathrm{concat}(\mathbf{x}_i, \mathbf{q})$.
 6: Finetune a LM $\mathtt{LM}_{\mathbf{\Theta}}$ using loss function in Eq. 2.
 7: Generate text embeddings: $\mathbf{e}_i = \mathtt{LM}_{\mathbf{\Theta}}(\mathbf{x}_i)$ (If LLM is used, replace $\mathbf{x}_i$ with $\mathbf{r}_i$).
 8: Train another GNN ($\mathtt{GNN}_{\hat{\mathbf{\Phi}}}$) on $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_{m+n}]$, $\mathbf{A}$, and $\mathbf{Y}_L$ with loss function in Eq. 1.
 9: Classify unlabeled nodes using $\mathtt{GNN}_{\hat{\mathbf{\Phi}}}$: $\overline{\mathbf{y}}_j = \mathrm{argmax}(\mathtt{GNN}_{\hat{\mathbf{\Phi}}}(\mathbf{e}_j, \mathbf{A})), j \in \{1, \cdots, n\}$.
10: return $\mathbf{Y}_U = [\overline{\mathbf{y}}_1, \cdots, \overline{\mathbf{y}}_n]$.
```

Table 7: Performance comparison of different models using complete labels.

Dataset	Cora(140)	Cora(1208)	PubMed(60)	PubMed(18217)
GCN	$0.6038_{\pm 0.0116}$	$0.7182_{\pm 0.0055}$	$0.6770_{\pm 0.0217}$	$0.8092_{\pm 0.0057}$
DeBERTa	$0.3048_{\pm 0.0198}$	$0.7366_{\pm 0.0137}$	$0.3616_{\pm 0.0976}$	$0.9510_{\pm 0.0056}$
DeBERTa+GPT	$0.5450_{\pm 0.1121}$	$0.7430_{\pm 0.0067}$	$0.4514_{\pm 0.0700}$	$0.9388_{\pm 0.0020}$
SFGL	$0.6314_{\pm 0.0191}$	$0.7278_{\pm 0.0151}$	$0.7238_{\pm 0.0124}$	$0.9382_{\pm 0.0078}$
SFGL+GPT	$0.6856_{\pm 0.0131}$	$0.7396_{\pm 0.0105}$	$0.8640_{\pm 0.0381}$	$0.9400_{\pm 0.0019}$

linear separation between points while disregarding the angular relationships or vector orientation, which are key factors in capturing semantic meaning.

Besides KNN graphs, our training framework can incorporate any LGI method to generate pseudo-labels. However, unlike KNN graphs, existing LGI methods typically require specialized optimization strategies with complex constraints and additional training on the entire dataset, leading to increased computational and model complexity. Moreover, ensuring that graphs generated by LGI methods exhibit a scale-free structure is challenging. For instance, SLAPS (Fatemi et al., 2021) requires several hours of training to generate graphs, whereas KNN graphs can be constructed in just a few minutes. Additionally, as shown in Fig. 8 (a), the graph produced by SLAPS does not exhibit scale-free properties, further highlighting the efficiency and structural advantages of KNN graphs.

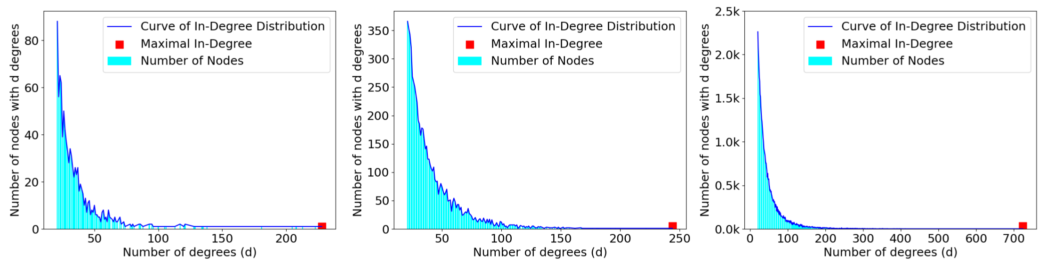
Additionally, we evaluate the performance of our models using the full set of available labels on the Cora and PubMed datasets. According to the standard data split, there are $1,068$ labeled samples in Cora and $18,157$ labeled samples in PubMed that remain unutilized during training. To assess the impact of complete supervision, we incorporate these additional labels, expanding the training sets to $1,208$ and $18,217$ labeled nodes for Cora and PubMed, respectively. Table 7 presents the results comparing the standard and complete label settings. With full supervision, the standalone DeBERTa model achieves strong performance, eliminating the need for pseudo-label generation, as the LMs already produce high-quality text embeddings. However, under the standard split with limited labeled nodes, generating pseudo-labels remains crucial for enhancing performance.

A.5 Training Strategies

The training algorithms of SFGL, SFGL+GPT, SFGL+$\underline{\text{GCN}}$ and SFGL+$\underset{\sim}{\text{GCN}}$ are outlined in Algorithms 1 and 2. The primary difference between SFGL and SFGL+GPT lies in steps 5 and 7, where SFGL+GPT incorporates GPT responses to enhance text embeddings. Similarly, the distinction between SFGL+$\underline{\text{GCN}}$ and SFGL+$\underset{\sim}{\text{GCN}}$ is that the former employs a LM to generate text embeddings, while the latter further refines them using an LLM. Specifically, SFGL+$\underset{\sim}{\text{GCN}}$ replaces the original node textual sequences with GPT-generated responses for finetuning. All these models use a GCN as the final classifier.

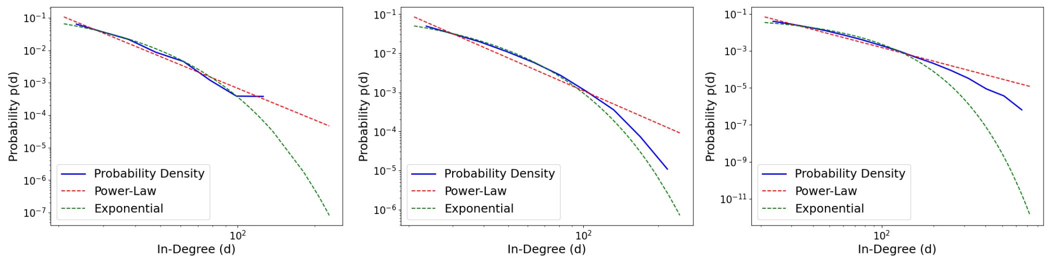
Algorithm 2 Scale-Free Graph-Language Model (SFGL+GCN / SFGL+GCN)

```
1: Return to step 1 of Algorithm 1, replace $\mathbf{F}$ with $\mathbf{E}$ generated in step 7.
2: Repeat steps 2, 3, and 4 of Algorithm 1 once to generate better pseudo labels.
3: Repeat steps 6, 7, 8, and 9 of Algorithm 1 once to re-classify the unlabeled nodes.
```



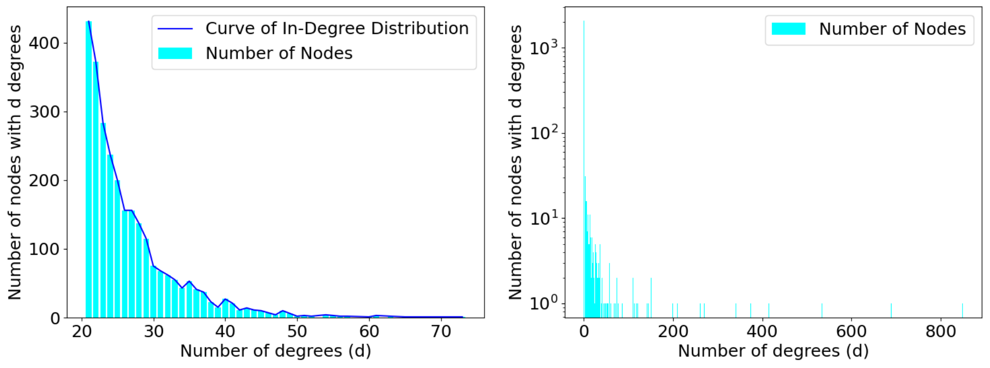
(a) Cora (b) Pubmed (c) ogbn-arxiv

Figure 6: In-degree distribution of KNN graphs on different datasets.



(a) Cora (b) Pubmed (c) ogbn-arxiv

Figure 7: Curve fitting to the in-degree distribution of KNN graphs on different datasets.



(a) SLAPS (b) Euclidean

Figure 8: In-degree distribution of graph generated by (a) SLAPS and (b) KNN (Euclidean).

A.6 Limitation

Due to the inherent differences in the formation mechanisms of KNN graphs and real-world networks, constructing a fully exact scale-free KNN graph remains challenging. A potential solution is to incorporate the scale-free property as a constraint within an optimization objective and then optimize it to learn an exact latent graph, which presents an intriguing direction for future research. Furthermore, this work primarily focuses on citation networks. Extending our findings and methodologies to other types of networks, such as social and biological networks, represents another promising avenue for future exploration.