Parameter-Efficient Tuning Large Language Models for Graph Representation Learning

Qi Zhu Amazon Web Services qzhuamzn@amazon.com

Shichang Zhang* University of California, Los Angeles shichang@cs.ucla.edu Da Zheng Amazon Web Services dzzhen@amazon.com

Bowen Jin* University of Illinois Urbana-Champaign bowenj4@illinois.edu

George Karypis Amazon Web Services gkarypis@amazon.com Xiang Song Amazon Web Services dzzhen@amazon.com

Yizhou Sun University of California, Los Angeles yzsun@cs.ucla.edu

ABSTRACT

Text-rich graphs, which exhibit rich textual information on nodes and edges, are prevalent across a wide range of real-world business applications. Large Language Models (LLMs) have demonstrated remarkable abilities in understanding text, which also introduced the potential for more expressive modeling in text-rich graphs. Despite these capabilities, efficiently applying LLMs to representation learning on graphs presents significant challenges. Recently, parameterefficient fine-tuning methods for LLMs have enabled efficient new task generalization with minimal time and memory consumption. Inspired by this, we introduce Graph-aware Parameter-Efficient Fine-Tuning - GPEFT, a novel approach for efficient graph representation learning with LLMs on text-rich graphs. Specifically, we utilize a graph neural network (GNN) to encode structural information from neighboring nodes into a graph prompt. This prompt is then inserted at the beginning of the text sequence. To improve the quality of graph prompts, we pre-trained the GNN to assist the frozen LLM in predicting the next token in the node text. Compared with existing joint GNN and LMs, our method directly generate the node embeddings from large language models with an affordable fine-tuning cost. We validate our approach through comprehensive experiments conducted on 8 different text-rich graphs, observing an average improvement of 2% in hit@1 and Mean Reciprocal Rank (MRR) in link prediction evaluations. Our results demonstrate the efficacy and efficiency of our model, showing that it can be smoothly integrated with various large language models, including OPT, LLaMA and Falcon.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

Representation Learning, Graphs, Language Models

ACM Reference Format:

1 INTRODUCTION

Information networks form the backbone of modern data systems: millions of daily social posts are shared among users on platforms like Facebook and Twitter; countless papers are published and cited within academic networks. Many of these networks are rich in textual information on various types of objects, known as text-rich or text-attributed networks. For example, in an e-commerce graph, text-rich information could be used to predict links by analyzing product descriptions, user reviews, and metadata to recommend items that are frequently bought together. Such real-world applications typify the problem of link prediction, where representation learning (*i.e.*, embedding) emerges as the most prevalent solution.

Representation learning on graph-structured data aims to learn a dense vector for each node through self-supervised tasks such as link prediction [9, 34, 46] or masked attribute recovery [14]. These dense vectors can be widely utilized in various downstream applications including search, ranking and retrieval. To effectively utilize both node attributes and structural information in representation learning, Graph Neural Networks (GNNs) [11, 21] devise a novel type of neural networks with message-passing mechanism. In text-rich graphs, raw text is usually transformed into feature vectors by a text encoder prior to the application of Graph Neural Networks (GNNs). Early text encoders, such as bag-of-words and Word2Vec[32], are being gradually phased out due to significant improvements in text representation from transformer-based language models (LMs).

This has spurred interest in jointly modeling textual and structural data with GNNs and LMs. Notable architectures include (1) a cascading architecture that integrates GNNs with LM features (cascading GNN-LMs); (2) nested GNN-LMs that blend message passing into the language modeling process. Examples of the latter

^{*}Work done while being an intern at Amazon.



Figure 1: Convenient use case of parameter-efficient finetuning for text-rich graphs, requiring only 2% additional LLM parameters per graph application.

include GraphFormer [44] and Heterformer [20], which introduce a nested GNN-transformer architecture for homogeneous and heterogeneous graph, respectively. Historically, GNN-LMs have focused on combining GNNs with medium-sized LMs such as BERT [4] and RoBERTa [27]. However, large language models (LLMs) [1, 42, 47] with billions of parameters, known for their exceptional multi-task generalization and instruction-following capabilities, present an untapped potential for further enhancing the performance of these models. Applying large language models (LLMs) to text-rich graphs presents an intriguing concept; however, fine-tuning such GNN-LLM models, as with existing architecures, is impractical due to the prohibitively high computational and memory demands. As an alternative, parameter-efficient fine-tuning (PEFT) techniques for LLMs, such as LORA [16] and Prefix Tuning [24], advocate for updating a minimal fraction of parameters (e.g., less than 1%) to achieve comparable performance. Inspired by this, our method integrates graph encoding into the PEFT framework, which enables efficient graph representation learning with LLMs.

Contrary to PEFT, another line of research [6, 45] explores incontext learning of LLMs without weight updates. These works transform the neighboring context of a target node into textual descriptions, incorporating features, and employ LLMs to make predictions such as node degrees, labels, and the existence of links. However, applying these methods to representation learning at an industrial scale presents notable challenges. These approaches are primarily tailored for tasks involving a limited number of candidates and do not account for the similarity between any pair of nodes. In a graph with *N* nodes, they require a substantial inference cost of $O(N^2)$ to iterate every positive edge, in contrast to the more efficient strategy of generating node embeddings at a cost of O(N)and subsequently calculating their similarities within the embedding space (see also the inference cost of in-context learning in Table 1). This disparity shows the limitation of in-context learning for text-rich graphs and emphasizes the need for more scalable methods to handle large-scale graph data efficiently.

In this work, we introduce Graph-aware Parameter-Efficient Fine-Tuning (*i.e.*, GPEFT) for **light-weighted** and **efficient** graph representation learning with LLMs. Our methodology leverages graph neural networks (GNNs) to encode neighboring nodes into a graph prompt, which is subsequently integrated into the original sequence as a "soft prompt". We employ a pre-trained LLM and update only the GNN prompt encoder and the PEFT parameters,

Table 1: Comparison of existing GNN-LMs for representation learning on a graph with one million edges. We use GPU hours to estimate the computation cost.

	GNN-LM [19]	LLM Service [45]	GPEFT
category	fine-tune	in-context learning	PEFT
#params per task	110M	-	14.4M
training cost	~\$250	-	~\$50
prediction cost	~\$5	>\$1000	~\$10

ensuring minimal computational cost. As shown in Figure 1, GPEFT does not change the original parameters of LLMs and only $\sim 2\%$ of additional parameters are optimized for each task on a given textrich graphs. For instance, for recommendations on social networks, we can simply load the SocialGraph::recommendation component into the backbone LLM and train GPEFT with task-specific supervision. To improve the quality of the graph prompt, we introduce a pre-training phase for the GNN prompt encoder. This phase utilizes a next-token prediction objective on node texts to align the distribution of graph prompt and word embeddings of LLMs. In the fine-tuning (*i.e.*, representation learning) phase, we choose to use contrastive loss [10] with one negative sample to avoid the memory overhead for traditional in-batch loss used in LM fine-tuning [8].

Compared with existing GNN-LMs, we reduce the training cost of GNN-LMs by parameter-efficient tuning and use structural information as input instead of a nested architecture; Compared with In-Graph Context Learning LLMs, we avoid the quadratic inference costs for link prediction. In Table 1, we calculate the approximated computation cost for these two methods and GPEFT. In our experiments, we conduct large-scale link prediction evaluation through representation learning across 8 different graphs in two domains. Our observations reveal an average improvement of 2% over large language models (LLMs) and 3% over the existing GNN-LMs. Our contributions are summarized as follows,

- We propose the first graph representation learning framework that utilizes Large Language Models with billions of parameters.
- We develop a novel Parameter-Efficient Fine-Tuning (PEFT) algorithm - GPEFT for text-rich graphs. This method effectively integrates the key structural information into LLMs as a graph prompt.
- Through comprehensive experiments conducted on eight text-rich graphs, we consistently observe improvements with our approach over previous GNN-LMs, while also achieving computational efficiency.

2 RELATED WORK

Modeling Text-Rich Graphs using LMs. Regarding the success of language models and GNNs in their respective areas, modeling text-rich graphs has been a hot topic[18]. (1) Graph-empowered GNN-LMs: GraphFormers [44] is a GNN-nested Transformer architecture that insert GNN modules between transformer layers. Using language models to model target and neighbor node texts requires huge memory and time costs. To address this, some work [28]

proposed freezing the language model to reduce the computation needed for cascading. Some work [20, 23] proposed neighbor sampling but that reduces the graph information captured. Therefore, recently some work propose to joint train LMs and GNNs through knowledge distillation [30] or Expectation Maximization algorithms [49]. (2) Self-supervised GNN-LMs: some methods [3, 30] directly supervise language model fine-tuning through graph-related tasks, to help language models better understand the textual information in text-attributed graphs. The language model is then combined with GNNs by freezing the language model. This approach demonstrates the inherent connections between graph structure and text in TAGs. However, current research in this direction has limitations in that the LM and graph are separate, and the language model cannot directly perceive graph information. It also does not utilize the inherent connections between language and graphs to help GNNs better learn structural features. (3) LLMs for Graph: With the breakthrough progress made by LLMs on textual tasks [1, 42], recently many works have emerged exploring how to directly utilize LLMs to understand text-attributed graphs [2]. Some works also explored using large models to enhance the textual features of text-attributed graphs [5, 13]. In terms of the representation learning, methods [6, 41, 45] that turns graph structure into in-context learning are most viable option for representation learning.

Parameter-Efficient Fine-Tuning of LLMs. As the size of language model continue to increase, full fine-tuning has became more and more impractical. Parameter-efficient fine-tuning (PEFT) [29] freeze most of the language model parameters, which has been successfully applied to popular language models such as BERT [4], GPT [1, 35, 36] and t5 [37]. This section reviews the key developments in the area. (1) Adapter Layers [15]: Adapter layers are small, trainable modules inserted between transformer layers of a pre-trained langauge model. (2) Prompt Tuning [22]: Prompt tuning leverages the pre-trained knowledge of LLMs by appending taskspecific prompts to the input text, effectively guiding the model's predictions without updating its parameters. (3) Low-Rank Adaptation(LORA) [16]: LORA modifies the attention and feedforward layers in the transformer through two low-rank updating matrices. These approaches can also be combined and found to be useful in applications like multi-modal instruction tuning models [7, 25].

3 NOTATIONS AND PRELIMINARIES

This section introduces the background knowledge, notations and technical terms that will be used throughout the paper.

3.1 Text-Rich Graph Representation Learning

Let $\mathcal{G}=(\mathcal{V},\mathsf{X},\mathcal{E})$ be defined as a text-rich graph with multiple edge types \mathcal{T} , where \mathcal{V} represents the set of nodes, X denotes the text features associated with nodes, and \mathcal{E} symbolizes the edges between nodes. Each node i is accompanied by a text sequence (e.g., item description, paper title and abstract) $\mathcal{S}_i = \{s_{i,0},...,s_{i,k}\}$. For each edge type $t \in \mathcal{T}$, an edge $e_{ij} \in \mathcal{E}_t$ indicates the presence of an edge between node i and j of type t. Given the target edge type t, we define the problem of representation learning as mapping nodes into the embedding space, considering all observed edges except those designated for testing, denoted by $\mathcal{E} \setminus \mathcal{E}_t^{\text{test}}$. The learning objective aims to maximize the likelihood of observing the training

edges $\mathcal{E}_t^{\text{train}}$. The evaluation of the learned node representations is conducted through link prediction on the testing edges $\mathcal{E}_t^{\text{test}}$.

3.2 Casual Language Modeling

Given a text sequence on each node, the causal language model (CLM) [35] aims to learn the probability distribution of these text sequences in a way that respects the sequential order of words or tokens. CLM is the most popular architecture for recent large language models such as GPT-3 [1] and LLaMA [42]. Specifically, the objective function of the CLM, $\mathcal{L}_{\rm LLM}$ is defined as the negative log-likelihood of predicting each subsequent token in the sequence given its preceding tokens. This training objective is also referred as "next token prediction" frequently in the literature.

$$\mathcal{L}_{\text{LLM}} = -\sum_{i} \sum_{j=0}^{k} \log P(s_{i,j+1} | s_{i,0}, \dots, s_{i,j}; \Theta_{\text{LLM}})$$
 (1)

In this equation, $P(s_{i,j+1}|s_{i,0},\ldots,s_{i,j};\Theta_{\rm LLM})$ represents the probability of generating the next token $s_{i,j+1}$ given the sequence of preceding tokens $(s_{i,0},\ldots,s_{i,j})$, parameterized by $\Theta_{\rm LLM}$. The architecture of a Large Language Model (LLM) is predominantly based on transformers [43]. This work explores the potential of using casual lanuagge model to learn the node representations in text-rich graphs.

Prompt Tuning. A prompt for LLMs is a text snippet concatenated with the original text sequence designed to bridge the gap between pre-training tasks (e.g., CLM) and downstream tasks, such as question answering or summarization. In prompt tuning, soft prompt denotes a method where learnable embeddings are utilized to steer the model's response [22, 26, 50] instead of supplying a concrete textual prompt. These soft prompts are typically concatenated with the input text data. Unlike fixed textual prompts, soft prompts are adaptable and are optimized to elicit the most effective response from the model. We model the structural information in $\mathcal G$ as a soft prompt in this work.

3.3 Parameter-Efficient LLM Fine-tuning

(1) Low-Rank Adaptation (LORA): Consider W as the original weight matrix in attention and feed-forward layer of the transformer. LoRA introduces two low-rank matrices A and B, where the product AB^T approximates the desired change in W. The modified weight matrix \hat{W} is then:

$$\hat{W} = W + AB^T \tag{2}$$

In this equation, the matrices A and B are trainable, while W remains fixed. The low-rank nature of AB^T ensures that the number of additional parameters is relatively small.

(2) Prefix-Tuning: Prefix Tuning involves adding a small number of trainable parameters (prefixes) to the input of each transformer layer. Let's denote the prefix as P and the original input as S. The modified input to the transformer layer can be represented as: [P; S].

4 METHOD

This section presents our proposed framework - GPEFT, which employs graph-aware PEFT techniques for representation learning in Large Language Models (LLMs). The core idea of GPEFT includes two key strategies: (1) integrating graph information through parameter-efficient tuning methods in LLMs, and (2) executing large-scale graph representation learning specifically for the task of link prediction.

Motivation: Language models accurately encode the textual information in a text-rich graph, many structure-related predictions, however, are difficult to infer from the text alone. As illustrated in Figure 2, some popular items in an e-commerce network or seminal papers in an academic graph are densely connected, even though their textual similarity to neighboring nodes varies significantly. Although several recent works [6, 13, 45] have integrated graph information into Large Language Models (LLMs), most focus on node or graph classification problems. However, the nature of link prediction differs significantly from classification, as it requires calculating the proximity between pairs of nodes. Motivated by this distinction, our work aims to learn node embeddings through graph-aware LLM fine-tuning.

Framework: The framework of GPEFT is shown in Figure 2. Given a text-rich graph \mathcal{G} , and target edge type for training, there are following three steps for representation learning: step 1: a GNN model called graph prompt encoder **prepend** the graph structural information before text sequence in the language model, we detail the descriptions in Section 4.1; step 2: To align the feature space of node representations output by GNN with the text embeddings in a LLM, we consider to use the casual language modeling to **pre-train** the GNN prompt encoder; step 3: an computationally efficient contrastive learning loss is employed to optimized the GNN-LLM model end-to-end with **parameter-efficient fine-tuning** techniques. We describe the pre-training and fine-tuning details in subsections of Section 4.2.

4.1 Graph prompt tuning

In the existing graph LLMs [6, 45], the graph encoding function Θ_g typically transforms the neighbors of node i into a textual sequence, such as "node_1: text, node_2: text, node_1 is connected with node_2…". However, this approach assumes that all neighbors are equally important to the target node, which can also result in very long input sequences for the LLM in a densely connected graph.

Structural representations. On the contrary, we transform the node texts x_i and structural information \mathcal{A} into dense vectors using GNNs. To achieve this, we fist use a small pre-trained langauge model - BERT [4] to turn the text sequence S_i into document embedding x_i . Then, we use a common message passing GNN like GCN [21] or GraphSAGE [11] to obtain the GNN embedding z_i for node i in the given graph,

$$z_i = \text{GNN}(x_i, X, \mathcal{A}; \Theta_q)$$
 (3)

where x_i is the center node feature, X and \mathcal{A} are neighbor node features and adjacency matrix, respectively. Θ_g represents the GNN parameters, which also contains a mapping function from the GNN

embedding space to the word embedding space of the LLM, \mathcal{M} : $\mathbb{R}^{d_{\text{GNN}}} \to \mathbb{R}^{d_{\text{model}}}$, and $d_{\text{GNN}} < d_{\text{model}}$. We also refer z_i as the (nodelevel) graph prompt in the remaining of the paper.

Prompting LLM with the graph prompt: With the newly created graph-aware representation of node z_i , we now discuss two different soft prompting strategies: (1) prepend or (2) append graph prompt to the target node sequence $\{s_{i,0},...,s_{i,k}\}$. We denote the hidden states in the transformer for j-th word in text sequence S_i as $h_{i,j}$. Following the prior research of representation learning using casual language model [33], we introduce a specialized graph representation learning [GRL] token at the end of the sequence to represent the final node representations. We opt to **prepend** the graph prompt because it offers greater expressive power, as detailed in the following theorem.

Theorem 4.1. Consider the l-th layer of a transformer, where $H_k^l = [h_{i,0}^l, \ldots, h_{i,k}^l]$ represents the embedding matrix for a text sequence of length k. At each layer, appending a graph prompt to the sequence results in a convex combination of the embeddings $[H_k^l; z_i]$. In contrast, prepending the graph prompt z_i to the sequence allows for the incorporation of z_i in the computation of H_k^l .

We observe that in the attention calculation, prepending the graph prompt can potentially enhance the model's expressiveness through direct interaction with the text sequence's embeddings. The detailed proof is provided in Appendix A.

4.2 Graph representation learning with LLMs

Following the previous section, we compute the node representation v_i as the last hidden representation of the special token [GRL] as follows:

$$v_i = \text{LLM}\left(\{z_i, s_{i,0}, s_{i,k}, [GRL]\}; \Theta_q, \Theta_{\text{LLM}}\right) \tag{4}$$

where Θ_g and $\Theta_{\rm LLM}$ are the trainable parameters of GNN and LLM. Now we discuss the motivation of pre-training GNN through casual language modeling.

Pre-training GNN Prompt Encoder. Note that node features x_i are independent to the large language model and the GNN Θ_g is randomly initialized. The feature distribution of the GNN prompt encoder $P(z_i)$ does not align with the word embeddings $P(h_i^0)$. We use the next token prediction objective to pre-train the GNN Θ_g while freezing the LLM Θ_{LLM} ,

$$\mathcal{L}_{\text{pre-training}} = -\sum_{i} \sum_{j=0}^{k} \log P(s_{i,j+1}|z_i, s_{i,0}, \dots, s_{i,j}; \Theta_g, \Theta_{\text{LLM}})$$
 (5)

where \mathcal{L} is the loss function, $P(s_{i,j+1}|z_i,s_{i,0},\ldots,s_{i,j};\Theta)$ is the conditional probability of the token $s_{i,j+1}$ given the GNN prompt z_i and the preceding tokens up to $s_{i,j}$, and Θ represents the trainable parameters of this step.

Fine-tuning. After pre-training, we use the cosine similarity between the GRL embeddings of node pairs to represent the likelihood of edges in the graph. In particular, we obtain the representations of each node v, compute the cosine distance $d_{ij} = 1 - \cos(v_i, v_j)$ between node i and j and employ a contrastive loss [10] between a

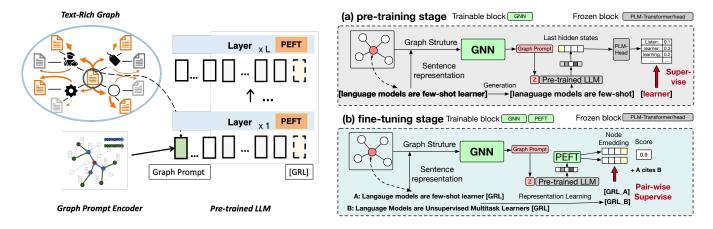


Figure 2: Overview of the GPEFT Framework: Architecture (Left) and Pre-training and Fine-tuning Processes (Right).

positive edge e_{ij} and random negative edge $e_{ij'}$:

$$\mathcal{L}_{\text{fine-tuning}} = \sum_{e_{ij} \in \mathcal{E}} \left(d_{ij}^2 + \max(\tau - d_{ij'}, 0)^2 \right)$$
 (6)

$$v_i = \text{LLM}\left(\{z_i, s_{i,0}, s_{i,k}, [GRL]\}; \Theta_g, \Theta_{\text{LLM}}\right)$$
 (7)

it encourages connected nodes to have similar representations while penalizes disconnected nodes when their cosine similarity is larger than the margin τ (i.e., $\tau=0.5$ in our experiment). In a LLM with billion parameters, usually the batch size cannot be large, hence we observe using such loss yields better performance and computational efficiency than traditional in-batch contrastive learning loss (e.g. SimCSE [8]) with a small batch size. Note that in this step, we update both the LLM and GNN parameters.

4.3 Model Optimization.

We outlined the training process of our framework in the previous section. However, it is well-known that directly optimizing a billion-scale LLM is impractical. Furthermore, maintaining a different set of full LLM parameters for various types of text-rich graphs can lead to space inefficiency. A more efficient approach would be the ability to store a small amount of parameters for a specific application related to a text-rich graph. To this end, we propose partitioning the parameters of the LLM into two segments: $\Theta_{\text{LLM}} = [\Theta_{\text{peft}}; \Theta_{\text{pre}}]$, where Θ_{pre} indicates the pre-trained weights of a specific LLM. This division allows **light-weighted** parameters for each specific application.

Now we describes how we integrates our graph prompt encoder Θ_g into traditional parameter efficient tuning Θ_{peft} while maintaining the representation power discussed before. We consider use two most popular peft variants:

- (1) Low-Rank Adaptation (LORA): In LORA, the trainable parameters are coupled in every transformer layer. Therefore, the naive prepending of the GNN prompt can effectively participate in the gradient updates of LORA matrices.
- (2) Prefix-Tuning: In prefix-tuning, the modified input to the transformer layer is typically represented as: [P; S]. However, placing graph tokens Z at the beginning of S does not

contribute to the updating of the prefix embedding P similar to the Theorem A.1. To address this, we propose a slight modification to the prefix embedding, denoted as P' = P + Z. This adjustment ensures that the graph information is effectively integrated with the prefix tuning process.

```
Algorithm 1: Pseudo code for GPEFT optimization
```

```
1 Input: text-rich graph \mathcal{G}, a set of training edges \mathcal{E}_t^{\text{train}}
 _{2}\; pre-trained LLM: \Theta_{LLM}, GNN encoder: \Theta_{q},
 3 Graph Neighborhood Sampler SAMPLE
 4 Output: node embeddings V for all nodes
 5 // Pre-training;
 6 for each batch of nodes \{x_i\} from SAMPLE(\mathcal{G}) do
         compute graph prompt z_i \leftarrow \text{Eq.}(3),
         causal language modeling \mathcal{L}_{\text{pre-train}} \leftarrow \text{Eq.}(5);
         update \Theta_q;
10 end
11 // Fine-tuning;
12 for each batch of edges \{e_{ij}\} from SAMPLE(\mathcal{G}, \mathcal{E}_t^{\text{train}}) do
         compute graph prompt z_i, z_j \leftarrow \text{Eq.}(3),
14
         \mathcal{L}_{\text{fine-tuning}} \leftarrow \text{Eq.}(6);
15
         update \{\Theta_q, \Theta_{peft}\};
16 end
17 // Evaluation;
18 for each batch of nodes \{x_i\} from SAMPLE(\mathcal{G}) do
        Inference node embedding v_i \leftarrow Eq.(4)
21 Compute evaluation metric on testing edges \mathcal{E}_t^{test}
```

In Algorithm 1, we detail the training procedure of GPEFT. In the pre-training phase, we iterate on every batch of nodes and sample neighbors of each node to compute the graph prompt at line 7. Line 8 and 9 correspond to the weight matrices updating of GNN prompt encoder Θ_g while freezing LLM. In the fine-tuning (the representation learning) phase, for every batch of training edges,

Table 2: Dataset Statistics.

Dataset	#Nodes	#Edges	#Task Edges	Avg Degree	Avg #Tokens
Clothing	469,274	2,578,746	1792,604	10.99	117.83
Home	453,121	3,732,948	2,382,201	16.48	133.94
Sports	293,712	2,390,076	3,130,128	16.27	125.08
VideoGames	38,869	729,062	844,296	37.51	147.33
Computer Science	263,393	2,929,358	13,188,472	22.24	159.69
Economics	178,670	3,064,144	3,615,524	34.3	160.21
Geology	431,834	15,256,890	35,915,142	70.66	205.08
Mathematics	490,551	4,770,644	12,911,644	19.45	143.94

we sample neighbors for both nodes and perform graph prompt encoding similarly. Then, we tune the parameters of PEFT and GNN prompt encoder simultaneously to minimize the representation loss (Line 14 and 15). In evaluation, we calculate the node embeddings as the [GRL] token in the sequence of each node and use the cosine similarity to rank the candidates edges.

Parameter Efficiency. We now describe the amount of additional parameters introduced in GPEFT. First, we use LORA [16] as an example to calculate the PEFT parameters. We denote the hidden dimension of GNN prompt encoder and LLM as $d_{\rm GNN}$ and $d_{\rm LLM}$, and rank of the LORA matrices as r, with $r << d_{\rm model}$. Assuming the LLM has L transformer layers, LORA introduces two matrices across four components of the transformer's feedforward network: query, key, value, and output. Consequently, the total additional parameters introduced by LoRA amount to $L \cdot 4rd_{\rm model}$. In the case of LLaMA, where $d_{\rm model} = 4096$ and L = 32, setting r = 16 yields approximately 8.9M additional parameters in $\Theta_{\rm peft}$.

For a k-layer GNN prompt encoder, setting $d_{\rm GNN}=768$ introduces $kd_{\rm GNN}^2 \cdot N_{\rm rel} + d_{\rm GNN}d_{\rm model}$ additional parameters, where $N_{\rm rel}$ represents the number of relations in ${\cal G}$. In our experiment, with $N_{\rm rel}=2, k=2$, this results in 5.5M parameters. Thus, we argue that our approach contains a comparable number of parameters to PEFT, which is significantly fewer than existing GNN-LMs (at least 110M for bert-base).

5 EXPERIMENTS

To evaluate the performance of GPEFT in learning representations of text-rich graphs, we focus on the following research questions (RQs) in this section:

- RQ1: How effective are large language models (LLMs) at graph representation learning, particularly with GPEFT?
- RQ2: Is the pre-training of the graph prompt encoder necessary?
- RQ3: Can our approach be adapted to different PEFT methods and LLMs?
- RQ4: Is GPEFT efficient in million-scale representation learning?
- RQ5: What is the parameter sensitivity of GPEFT?

5.1 Datasets

Some of the most popular benchmarks, such as OGB [17], do not include full-text information for link prediction tasks. Following recent research trends [19, 48], we benchmark on two representative text-rich graphs from the academia and e-commerce domains.

In each domain, we choose four different subdomains to comprehensively study the performance of link prediction with different methods.

Amazon Review [12, 31]. The Amazon Products dataset comprises commercial data from items sold on Amazon, complete with detailed descriptions. In this dataset, items are represented as nodes in a graph, with edges connecting pairs of nodes if they are *coviewed* or *co-purchased* by users. We utilize the *co-viewed* edges as observed data and perform our representation learning evaluation on the *co-purchased* edges, as this closely resembles real-world product recommendation scenarios. Our focus is on products within four specific subdomains: Clothing, Home, Sports, and Video Games. Each of these subdomains features a graph with over 1 million edges.

Microsoft Academia Graph [39] (MAG). MAG is a large, heterogeneous graph containing scientific publication records, citation relationships between those publications, as well as authors, institutions, journals, conferences, and fields of study. We use a pre-processed version (MAPLE [48]) to obtain field specific textrich graphs, that are Computer Science, Economics, Mathematics, Geology and *etc.*. We utilize the *cite-by* edges as observed data and generate paper pairs that share the same author (*same-author*) using author information of each paper.

We summarize the detailed data statistics of each subdomains in Amazon Review and MAG, including average node degrees and text length in Table 2.

5.2 Baselines

In our experiment, we compare GPEFT with different state-of-theart text-rich graph modeling algorithms (<u>underlined</u> in the following paragraphs).

Cascaded GNN-LMs: We consider methods that employ fine-tuned language models (LMs) or large language models (LLMs) as feature encoders, followed by training a graph neural network (GNN) on these features. Specifically, we select MPNET [40] that is a variant of Sentence-BERT [38] and LLaMA fine-tuned with LORA techniques. The LMs or LLMs are not updated in these baselines.

Graph-empowered LMs: We opt to compare with GraphFormers [44], which are designed for graph representation learning by aggregating the [CLS] tokens of neighboring nodes between transformer layers. Similar to our graph prompt approach, this method introduces virtual structure tokens to the LM. Patton [19] improves upon GraphFormers by incorporating pre-training with masked token and node prediction objectives. Additionally, we also report the performance of fine-tuning Sentence-BERT [38] without graph information.

Graph-aware LLMs: These methods do not alter the architectures of large language models (LLMs) but rather engage in in-context learning or fine-tuning of the LLMs. In the context of link prediction, the most suitable algorithm is <u>InstructGLM</u> [45], which translates both the graph structure and node features into natural language. We have made minor modifications to the InstructGLM pipeline, notably by utilizing a public LLM API¹ to transform the composed

 $^{^{1}} text-embedding-3-small\ in\ https://platform.openai.com/docs/guides/embeddings$

Table 3: Performance of link prediction on Amazon Review Graph. Each experiment is repeated three times, except for † : We limit API calls to just once for cost efficiency.

Method	Clothing		Home&Kitchen		Sports		Video Games		Average	
1110111011	Hit@1	MRR	Hit@1	MRR	Hit@1	MRR	Hit@1	MRR	Hit@1	MRR
GNN (Sentence-BERT) GNN (PEFT-LLaMA)	74.52 _{0.52} 76.22 _{0.26}	82.58 _{0.4} 84.16 _{0.19}	74.18 _{0.27} 73.74 _{7.03}	82.67 _{0.21} 81.66 _{6.43}	61.2 _{0.12} 62.26 _{0.43}	74.48 _{0.16} 75.36 _{0.91}	52.64 _{0.15} 56.14 _{0.18}	68.53 _{0.13} 71.59 _{0.14}	65.64 67.09	77.07 78.19
Sentence-BERT GraphFormers PATTON	62.11 _{0.20} 71.30 _{0.04} 76.95 _{0.02}	73.36 _{0.14} 79.31 _{0.02} 83.79 _{0.01}	65.43 _{0.17} 74.29 _{0.03} 78.14 _{0.05}	76.37 _{0.14} 81.63 _{0.02} 84.72 _{0.03}	50.13 _{0.13} 58.35 _{0.01} 62.44 _{0.02}	66.28 _{0.10} 71.45 _{0.00} 74.62 _{0.01}	41.83 _{0.05} 49.67 _{0.04} 51.07 _{0.14}	59.65 _{0.10} 65.98 _{0.04} 66.97 _{0.08}	54.88 63.40 67.15	68.92 74.60 77.53
InstructGLM-embeddings [†]	76.23	82.60	79.82	85.93	62.50	73.25	48.18	63.00	66.68	76.20
PEFT-LLaMA GraphPEFT w.o. pretraining GraphPEFT	74.73 _{0.02} 76.74 _{0.07} 76.95 _{0.07}	82.87 _{0.00} 84.57 _{0.06} 84.71 _{0.06}	78.93 _{0.02} 79.68 _{0.18} 79.87 _{0.18}	86.07 _{0.02} 86.63 _{0.09} 86.76 _{0.09}	62.52 _{0.02} 64.44 _{0.14} 64.61 _{0.14}	75.77 _{0.01} 77.21 _{0.12} 77.34 _{0.12}	56.07 _{0.18} 50.60 _{0.11} 58.04 _{0.11}	71.54 _{0.14} 66.97 _{0.14} 73.07 _{0.14}	68.06 67.87 69.88	79.06 78.85 80.47

Table 4: Performance of link prediction on MAG graph. Each experiment is repeated three times, except for † : We limit API calls to just once for cost efficiency.

Method	Computer Science		Economics		Geology		Mathematics		Average	
1/101101	Hit@1	MRR	Hit@1	MRR	Hit@1	MRR	Hit@1	MRR	Hit@1	MRR
GNN (Sentence-BERT)	24.97 _{0.15}	41.18 _{0.25}	28.64 _{0.13}	43.35 _{0.3}	35.94 _{0.22}	51.28 _{0.2}	47.67 _{0.21}	62.24 _{0.18}	34.31	49.51
GNN (PEFT-LLaMA)	27.81 _{0.10}	44.92 _{0.13}	31.89 _{0.45}	47.83 _{0.37}	35.68 _{0.12}	51.59 _{0.13}	46.24 _{0.11}	61.9 _{0.03}	35.41	51.56
Sentence-BERT	20.45 _{0.05}	36.64 _{0.03}	22.65 _{0.12}	37.39 _{0.14}	29.64 _{0.21}	45.37 _{0.13}	35.85 _{0.21}	52.23 _{0.18}	27.15	42.91
GraphFormers	19.14 _{0.09}	34.08 _{0.05}	19.86 _{0.06}	32.66 _{0.02}	29 _{0.03}	43.65 _{0.03}	39.91 _{0.13}	54.50 _{0.10}	26.98	41.22
PATTON	21.68 _{0.12}	36.48 _{0.07}	31.07 _{0.06}	44.99 _{0.05}	33.38 _{0.13}	48.06 _{0.07}	43.35 _{0.07}	57.37 _{0.04}	32.37	46.73
InstructGLM-embeddings [†]	21.84	34.88	24.76	36.95	34.55	48.00	43.01	55.22	31.04	43.76
PEFT-LLaMA GraphPEFT w.o. pretraining GraphPEFT	28.45 _{0.62}	45.16 _{0.91}	33.36 _{0.07}	49.07 _{0.14}	37.03 _{0.47}	52.56 _{0.69}	45.7 _{0.25}	61.34 _{0.15}	36.14	52.03
	23.86 _{0.19}	40.11 _{0.29}	27.19 _{0.18}	41.85 _{0.18}	34.93 _{0.25}	50.37 _{0.22}	45.81 _{0.00}	60.75 _{0.17}	32.95	48.27
	30.41 _{0.29}	47.37 _{0.25}	35.42 _{0.10}	51.08 _{0.13}	39.69 _{0.15}	55.19 _{0.14}	48.09 _{0.06}	63.02 _{0.01}	38.40	54.17

sequence into dense embeddings. Our adapted approach is referred to as InstructGLM-embeddings.

At last, we design several ablations of our model to verify the effectiveness of each component: (1) GPEFT without pre-training phase in Equation 5; (2) GPEFT without graph prompt, namely, PEFT.

For different methods, we use the same neural network architecture for GNN, LM and LLM for fair comparison. Specifically, we use a 2-layer GraphSAGE [11] as the graph neural network with hidden dimension $d_{\rm GNN}$ as 768, sentence transformer (*i.e.*, MPNet-v2 [40]) as the pre-trained language model and LLaMA [42] as the default pre-trained foundation model.

5.3 Experiment Settings

We conduct a large-scale representation learning experiment on text-rich graphs for link prediction. Specifically, we simulate real-world scenarios supporting tasks such as item recommendation through *co-purchase* prediction on Amazon Review and authorship identification through *same-author* prediction on MAG. In each subdomain, we randomly select 50,000 edges for training, 10,000 edges for validation, and use the remaining edges for testing. Following the same setup as in OGB [17], we select 100 negative edges for each test edge and compute the average hit@1 and MRR score across

all test edges. For all baseline methods, except GraphFormers and Patton, we employ the same contrastive loss as of Equation 6. Additionally, we standardize the neighborhood fan-outs (i.e., 5 neighbors at each hop) for GNNs.

We implement all baselines using Deep Graph Library (https://www.dgl.ai/) and Hugging Face (https://huggingface.co/). For all methods, we train them on 50K training samples for 4 epochs on 8 Nvidia A100 GPUs with a total batch size of 32. We set the peak learning rate as 1e-4. More details can be found in Appendix B.

5.4 Experiment Results

We compare our approach with state-of-the-art baselines across two different domains to assess the effectiveness of our representation learning method. We conduct three runs for each method and report the mean and standard deviation for each in Tables 1 and 2, resulting in the following observations:

RQ1: How effective are large language models (LLMs) at graph representation learning, particularly with GPEFT?

(1) Although masked language models (e.g., Sentence-BERT) were popularly used for sequence representation, large language models (e.g., PEFT-LLaMA) have shown improved performance over masked language models. In our experiments, LLMs yield more than 10% improvement over Sentence-BERT.

(2) In the same tables, we also find GNN (Sentence-BERT) perform worse than GNN (PEFT-LLaMA). In addition, GraphFormers also reports subpar performance against GNN (PEFT-LLaMA). Therefore, we conclude that large language models serve as powerful feature encoders for representation learning on text-rich graphs. Meanwhile, InstructGLM outperforms GraphFormers but falls short of PATTON's performance. This discrepancy may be attributed to the challenges of handling graph structures through embedding-based in-context learning, which struggles with accurately capturing structures via natural language descriptions. Consequently, carefully fine-tuned GNN-LMs (GPEFT and PATTON) tend to surpass the embeddings produced by black-box LLMs.

(3) GPEFT consistently outperforms the compared baseline across both metrics. Specifically, it surpasses PEFT-LLaMA by 2.6% and 1.8% on MAG and Amazon Reviews, respectively, in terms of hit@1. Compared to its own ablation, we find that pre-training helps improve the accuracy of link prediction, especially in the academic text-rich graph.

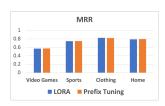
RQ2: Is the pre-training of the graph prompt encoder necessary?

On MAG, we observe that our approach performs significantly worse ($\sim 6\%$) without pretraining compared to its pretrained counterpart. On Amazon Review, while the performance gap is smaller across three subdomains, the Video Games category demonstrates instability without pretraining. Interestingly, Patton also outperforms GraphFormers with the same architecture on subdomains that utilize pre-trained checkpoints. Both methods introduce virtual structural tokens, and employing the same pre-training objective as the pre-trained language model. Apparently, continual training helps align the virtual token representations with text representations. Therefore, we advocate for a pre-training then fine-tuning paradigm, where one needs to pre-train only once in a domain and can fine-tune for various applications (e.g., co-purchase, churn, etc.).

5.5 Model Analysis

In this section, we provide more in-depth study to understand performance and efficiency of GPEFT.

RQ3:Can our approach be adapted to different PEFT methods and LLMs?





(a) Varying PEFT algorithms

(b) Varying backbone LLMs

Figure 3: GPEFT using different LLMs and PEFT algorithms on Amazon Review.

We first apply prefix tuning [24] on GPEFT and demonstrate the results on four subdomains of Amazon Review in Figure 3a. We can find that using prefix-tuning or LORA does not show much

Table 5: Time and memory costs of GPEFT on MAG-Economics using 8 A100 GPUs with a total batch size of 32.

	pre-training	fine-tuning	inference
Time	36min	96min	11min
Memory	20883MB	28309MB	13676MB
#trainable	5.5M	14.4M	-

difference, which indicates the versatility of our framework on various PEFT techniques.

Second, we substitute the backbone language model with OPT-1.3B [47] and Falcon-7B². In Figure 3b, we observe that, in general, LLaMA-7B outperforms the others, with Falcon-7B in second place and OPT-1.3B being outperformed by these two models. Our results are consistent with other work that shows LLMs with more parameters yield better performance in downstream applications.

RQ4: Is GPEFT efficient in large-scale representation learning?

In Table 5, we report the running time of our approach at each phase on MAG-Economics. Specifically, unlike the pre-training time of PATTON [19] (10hr+), the pre-training time of GPEFT is even shorter than the fine-tuning time. Because we only optimize Θ_g in the pre-training phase and we can say that the pre-training is both necessary and efficient together with **RQ2**. Owing to its parameter-efficient tuning, our approach not only minimizes the number of training parameters but also achieves the best performance. As mentioned in the introduction, the design of GPEFT aims to provide a high-quality, lightweight framework suitable for various applications on different text-rich graphs. Considering the reduced parameter storage requirements and the training time, we believe GPEFT holds significant potential for industrial-scale applications, including recommendation systems, ranking tasks, and etc.

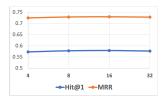
RQ5: What is the parameter sensitivity of GPEFT?

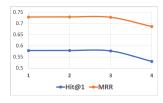
There are several important hyper-parameters in GPEFT: (1) LORA rank r, it affects the number of trainable parameters (2) number of hops k in GNN prompt encoder, it affects the amount of structural information used in the training of GPEFT. In Figure 4, we observe that varying the rank of LoRA matrices from 4 to 32 does not significantly affect the performance. This observation aligns with findings from other studies, which suggest that optimizing for a single task requires only minor adjustments to the pre-trained LLM weights. Similarly, varying number of hops in the GNN prompt encoder has a minor effect on performance, until the point where message passing begins to aggregate more noisy neighbors than useful ones (e.g., at 4 hops, as shown in Figure 4b).

6 CONCLUSION AND FUTURE WORK

This paper proposes GPEFT to harness LLMs for representation learning on text-rich graphs. Compared with existing work of applying LLM on graph structure data, our proposed method is the first one that generates embeddings from the LLM and therefore can be applied on numerous industrial applications. More importantly,

²https://huggingface.co/tiiuae/falcon-7b





- (a) Varying LORA rank
- (b) Varying number of hops

Figure 4: GPEFT using different LLMs and PEFT algorithms on Amazon Review.

GPEFT only requires a small amount of training parameters and extra storage for obtained models. Across eight different link prediction experiments across two domains, GPEFT improves hit@1 and mrr by 3% over the second best baselines. We validate the effectiveness of GPEFT with different peft methods and mutiple LLMs such as OPT, LLaMA and Falcon.

REFERENCES

- 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. NeurIPS (2020).
- [2] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2023. Exploring the potential of large language models (llms) in learning on graphs. arXiv preprint arXiv:2307.03393 (2023).
- [3] Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Ol-gica Milenkovic, and Inderjit S Dhillon. 2021. Node feature extraction by self-supervised multi-scale neighborhood prediction. arXiv preprint arXiv:2111.00064 (2021).
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. <u>arXiv</u> preprint arXiv:1810.04805 (2018).
- [5] Keyu Duan, Qian Liu, Tat-Seng Chua, Shuicheng Yan, Wei Tsang Ooi, Qizhe Xie, and Junxian He. 2023. Simteg: A frustratingly simple approach improves textual graph learning. arXiv preprint arXiv:2308.02565 (2023).
- [6] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a Graph: Encoding Graphs for Large Language Models. In NeurIPS 2023 Workshop: New Frontiers in Graph Learning. https://openreview.net/forum?id=7CAJpRo1Q8
- [7] Peng Gao, Jiaming Han, Renrui Zhang, Ziyi Lin, Shijie Geng, Aojun Zhou, Wei Zhang, Pan Lu, Conghui He, Xiangyu Yue, et al. 2023. Llama-adapter v2: Parameter-efficient visual instruction model. arXiv preprint arXiv:2304.15010 (2023)
- [8] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. arXiv preprint arXiv:2104.08821 (2021).
- [9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 855–864.
- [10] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06), Vol. 2. IEEE, 1735–1742.
- [11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. <u>Advances in neural information processing systems</u> 30 (2017).
- [12] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In <u>proceedings</u> of the 25th international conference on world wide web. 507–517.
- [13] Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2023. Harnessing Explanations: LLM-to-LM Interpreter for Enhanced Text-Attributed Graph Representation Learning. arXiv:2305.19523 [cs.LG]
- [14] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 594–604.
- [15] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In <u>International Conference on Machine Learning</u>. PMLR, 2790–2799.

- [16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021).
- [17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. arXiv preprint arXiv:2005.00687 (2020).
- [18] Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. 2023. Large Language Models on Graphs: A Comprehensive Survey. <u>arXiv preprint</u> arXiv:2312.02783 (2023).
- [19] Bowen Jin, Wentao Zhang, Yu Zhang, Yu Meng, Xinyang Zhang, Qi Zhu, and Jiawei Han. 2023. Patton: Language Model Pretraining on Text-Rich Networks. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics.
- [20] Bowen Jin, Yu Zhang, Qi Zhu, and Jiawei Han. 2022. Heterformer: Transformer-based Deep Node Representation Learning on Heterogeneous Text-Rich Networks. arXiv preprint arXiv:2205.10282 (2022).
- [21] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016).
- [22] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691 (2021).
- [23] Chaozhuo Li, Bochen Pang, Yuming Liu, Hao Sun, Zheng Liu, Xing Xie, Tianqi Yang, Yanling Cui, Liangjie Zhang, and Qi Zhang. 2021. Adsgnn: Behavior-graph augmented relevance modeling in sponsored search. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 223–232.
- [24] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190 (2021).
- [25] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. arXiv preprint arXiv:2304.08485 (2023).
- [26] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 61–68.
- [27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
- [28] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2019. Fine-grained fact verification with kernel graph attention network. <u>arXiv preprint arXiv:1910.09796</u> (2019).
- [29] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft.
- [30] Costas Mavromatis, Vassilis N Ioannidis, Shen Wang, Da Zheng, Soji Adeshina, Jun Ma, Han Zhao, Christos Faloutsos, and George Karypis. 2023. Train Your Own GNN Teacher: Graph-Aware Distillation on Textual Graphs. arXiv preprint arXiv:2304.10668 (2023).
- [31] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval. 43–52.
- [32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In Neural and Information Processing System (NIPS). https://papers.nips.cc/paper/5021-distributed-representations-of-wordsand-phrases-and-their-compositionality.pdf
- [33] Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. arXiv preprint arXiv:2202.08904 (2022).
- [34] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 701–710.
- [35] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).
- [36] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. OpenAI blog (2019)
- [37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. <u>Journal of Machine</u> Learning Research 21, 140 (2020), 1–67. http://jmlr.org/papers/v21/20-074.html
- [38] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084 (2019).
- [39] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. 2015. An overview of microsoft academic service (mas) and applications. In Proceedings of the 24th international conference on world wide web. 243–246.
- [40] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. <u>Advances in</u>

- Neural Information Processing Systems 33 (2020), 16857–16867.
- [41] Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V Chawla, and Panpan Xu. 2023. Graph Neural Prompting with Large Language Models. arXiv preprint arXiv:2309.15427 (2023).
- [42] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. <u>arXiv</u> preprint arXiv:2302.13971 (2023).
- [43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems 30 (2017).
- [44] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. 2021. GraphFormers: GNN-nested transformers for representation learning on textual graph. <u>Advances in Neural</u> Information Processing Systems 34 (2021), 28798–28810.
- [45] Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs. arXiv preprint arXiv:2308.07134 (2023).
- [46] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. 2019. Heterogeneous graph neural network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 793–803.
- [47] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. <u>arXiv preprint arXiv:2205.01068</u> (2022).
- [48] Yu Zhang, Bowen Jin, Qi Zhu, Yu Meng, and Jiawei Han. 2023. The effect of metadata on scientific literature tagging: A cross-field cross-model study. In WWW'23. 1626–1637.
- [49] Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. 2022. Learning on large-scale text-attributed graphs via variational inference. arXiv preprint arXiv:2210.14709 (2022).
- [50] Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [mask]: Learning vs. learning to recall. arXiv preprint arXiv:2104.05240 (2021).

A THEORETICAL ANALYSIS

Theorem A.1. Consider the l-th layer of a transformer, where $H_k^l = [h_{i,0}^l, \ldots, h_{i,k}^l]$ represents the embedding matrix for a text sequence of length k. At each layer, appending a graph prompt to the sequence results in a convex combination of the embeddings $[H_k^l; z_i]$. In contrast, prepending the graph prompt z_i to the sequence allows for the incorporation of z_i in the computation of H_k^l .

PROOF. We first provide the calculation of two different position of graph prompt token in a casual language model.

Prompt token after text sequence. In the l-th transformer layer, let us denote $H_j^l = [h_{i,0}^l, \ldots, h_{i,j}^l]$ as the embedding matrix for the first j tokens. Then, the attention calculation for the (k+1)-th hidden states $h_{i,j}^{l+1}$, incorporating the prompt token z_i , is expressed as follows:

Attention
$$(h_{i,j}^{l+1}) = \operatorname{softmax}\left(\frac{Q(h_{i,j}^l)K(H_j)^T}{\sqrt{d_k}}\right)V(H_j^l)$$
 (8)

Interestingly, in a causal language modeling setup, the attention scores for text tokens remain unchanged when a graph prompt is appended afterwards. This is because, in causal language modeling, a token does not have visibility of subsequent tokens. As a result, the hidden representation of H_k^I is the same before and after appending the prompt token. The representation of <code>[GRL]</code> is $h_{\text{GRL}} = \mathbf{a}^T [H_k^I; z_i]$, $\sum a_j = 1$, which is a convex combination of original hidden representation and graph prompt z_i . This is equivalent as cascading GNN-LMs that language model and graph neural network calculations are independent with each other.

Prompt token before text sequence. When we place graph prompt in the beginning, z_i^l is included in the attention score calculation of every token:

$$\operatorname{Attention}(h_{i,j}^{l+1}) = \operatorname{softmax}\left(\frac{Q(h_{i,j}^{l})K([z_{i}^{l}; H_{j}^{l}])^{T}}{\sqrt{d_{k}}}\right)V([z_{i}^{l}; H_{j}^{l}]) \tag{9}$$

In this manner, the hidden representation of text tokens, denoted as $\widehat{H_k^l} \neq H_k^l$, incorporates the information of z_i because the attention mask does not exclude the prompt token. Consequently, prepending the graph prompt employs the LLM as the predictor, whereas appending the prompt utilizes the GNN as the predictor.

B DETAILED EXPERIMENT SETTINGS

We use the PEFT library (https://github.com/huggingface/peft) from Hugging Face to implement the LORA and Prefix Tuning of LLMs. The graph prompt z_i is inserted at the beginning of the input embedding within the forward function of Hugging Face's AutoModel, and a "1" is prepended to the attention mask.

The hyper-paramters of GPEFT are shown in Table 6.

Table 6: Hyperparameters of GPEFT.

parameter name	value
pre-training batch size B	8
pre-training learning rate lr	1e-3
fine-tuning size B	4
fine-tuning learning rate lr	1e-4
GNN hidden dimension $d_{ m GNN}$	768
LLM hidden dimension $d_{ m modem}$	4096
maximal sequence length N	256
maximal gradient norm	1.0
neighborhood size K	5
# GNN layers L	2
# warmup epochs	1
# epochs	4
# random seeds	3