Prompt Tuning on Graph-augmented Low-resource Text Classification

Zhihao Wen and Yuan Fang *Senior Member, IEEE*

Abstract—Text classification is a fundamental problem in information retrieval with many real-world applications, such as predicting the topics of online articles and the categories of e-commerce product descriptions. However, low-resource text classification, with no or few labeled samples, presents a serious concern for supervised learning. Meanwhile, many text data are inherently grounded on a network structure, such as a hyperlink/citation network for online articles, and a user-item purchase network for e-commerce products. These graph structures capture rich semantic relationships, which can potentially augment low-resource text classification. In this paper, we propose a novel model called Graph-Grounded Pre-training and Prompting (G2P2) to address low-resource text classification in a two-pronged approach. During pre-training, we propose three graph interaction-based contrastive strategies to jointly pre-train a graph-text model; during downstream classification, we explore handcrafted discrete prompts and continuous prompt tuning for the jointly pre-trained model to achieve zero- and few-shot classification, respectively. Moreover, we explore the possibility of employing continuous prompt tuning for zero-shot inference. Specifically, we aim to generalize continuous prompts to unseen classes while leveraging a set of base classes. To this end, we extend G2P2 into G2P2*, hinging on a new architecture of conditional prompt tuning. Extensive experiments on four real-world datasets demonstrate the strength of G2P2 in zero- and few-shot low-resource text classification tasks, and illustrate the advantage of G2P2* in dealing with unseen classes.

Index Terms—Text classification, graph, low-resource learning, pre-training, prompt.

I. INTRODUCTION

Text classification is a fundamental research problem with many important applications in information retrieval. For example, predicting the topics of online articles can help readers easily search and navigate within the website or portal [1], and classifying the category of e-commerce product descriptions enables businesses to structure their inventory efficiently and improve users' search experience [2]. Recent advances in natural language processing (NLP) have achieved remarkable success for text classification, especially when there are large-scale and high-quality labeled data. However, data labeling is often costly and time-consuming, making low-resource classification, in which no or few labeled samples are available, an appealing alternative.

To address low-resource text classification, one approach is to utilize pre-trained language models (PLM) [3], [4], many of which are based on the transformer architecture [5] due to its powerful ability of encoding texts. A PLM can be adapted

Zhihao Wen and Yuan Fang (corresponding author) are with School of Computing and Information Systems, Singapore Management University, Singapore, 188065.

E-mail: zhwen.2019@phdcs.smu.edu.sg and yfang@smu.edu.sg

to different tasks by *fine-tuning* the model parameters to task-specific objectives. While the "pre-train, fine-tune" paradigm requires fewer labeled data than traditional supervised learning, it suffers from two drawbacks. Firstly, state-of-the-art PLMs typically have huge model size, *e.g.*, GPT-3 has 175 billion parameters [6], which makes fine-tuning prohibitively expensive [7]. Secondly, fine-tuning still needs a reasonable amount of labeled data due to the gap between pre-training and fine-tuning objectives, and thus struggles with low-resource scenarios including zero- and few-shot classification.

To overcome the problem of pre-training and fine-tuning, *prompting* [6] has been proposed. It uses a natural language instruction or "prompt" to give a hint of the downstream task, whilst freezing the parameters of a large PLM. In other words, no fine-tuning or additional training is required at all for a new task. However, discrete natural language prompts can be difficult to design and may result in suboptimal performance compared to fine-tuning [8]. More recently, *prompt tuning* [8], [9] formulates a continuous prompt as a learnable embedding, which is optimized without updating the PLM.

Meanwhile, text data frequently rely on network structures, such as hyperlink/citation networks for online articles or user-item interaction graphs for e-commerce products. These graph structures expose valuable relationships between content or descriptions, aiding low-resource text classification. While current PLMs and prompting do not leverage these relationships, graph neural networks (GNNs) [10] excel in processing graph data. GNNs typically leverage a message-passing architecture, which allows for the integration of node features and topological structures, resulting in impressive performance on graphs. Nevertheless, traditional end-to-end training of GNNs heavily relies on abundant task-specific labels, which motivates self-supervised GNNs [11] using well-designed pretext tasks derived from a label-free graph in a contrastive [12] or generative [13], [14] manner. However, the treatment of text features in GNNs remains rudimentary. A simple bag-of-words representation [15] or aggregation of shallow word embeddings [16] is fed into GNNs as the "initial message", which are further propagated along graph structures. Hence, the modeling of texts is coarse-grained, unable to fully capture the subtle semantic differences and similarities within texts.

Present work: G2P2. To overcome the limitations of existing text- and graph-based solutions, we propose a novel approach called Graph-Grounded Pre-training and Prompting (**G2P2**). G2P2 attempts to address the following two open questions.

Firstly, *how do we capture fine-grained textual semantics, while leveraging graph structure information jointly?* A naïve

approach is to use a language model to generate features from raw texts as input, and then train a GNN. However, in this way the texts and graph are only loosely coupled, lacking an explicit pairing to complement each other. In this paper, we propose graph-grounded contrastive pre-training, to maximize the alignment between text and graph representations based on three types of graph interaction, namely, text-node, text-summary, and node-summary interactions.

Secondly, *how do we augment low-resource text classification given a jointly pre-trained graph-text model?* Instead of following the traditional fine-tuning paradigm, we try to "prompt" our jointly pre-trained graph-text model, from which the most relevant structural and semantic information can be located to improve low-resource classification. Without the need to update a large pre-trained model, prompting is also more efficient than fine-tuning. Specifically, we employ discrete prompts in zero-shot classification and continuous prompts in few-shot settings. While discrete prompts are manually crafted in the absence of class labels, continuous prompts can be automatically learned from the few-shot labels through a prompt-tuning process. On one hand, discrete prompts (**G2P2+d**) often require intensive human labor, and the performance of tuned prompts can be uneven across tasks. On the other hand, continuous prompt leverages the inherent differentiability of neural networks to tune the optimal prompts, requiring minimal human input. Prompt-tuning is both data- and computation-efficient owing to the much fewer parameters in a continuous prompt than in the pre-trained model. Furthermore, considering the graph structures between texts, we propose a context-based initialization for prompt tuning, which could provide a more informative starting point.

Extension: G2P2*. While continuous prompt performs better and requires less human effort than discrete prompts, tuning continuous prompts still need some labeled data and thus would not work for zero-shot inference. Thus, let us consider an alternative zero-shot classification setting with both *base classes* and *unseen classes*: Each base class includes a small labeled set of instances for selecting discrete prompts (G2P2+d) or tuning continuous prompts (G2P2), while each unseen class has no labeled set.

Under this new zero-shot setting, continuous prompts can be tuned on the base classes, but the tuned prompts tend to overfit to the base classes and fall short in extending their reach to broader, unseen classes. For example, in Fig. 1(a), while G2P2 demonstrates strong performance on the base classes like "ink" or "pencils", its performance declines sharply when applying the tuned prompts on the unseen classes like "oil paint" or "canvas". The significant drop in performance is primarily attributed to the potential class shifts often present even within the same domain, which hinders G2P2's ability to generalize effectively to unseen classes. Therefore, an important research question emerges: *How to generalize continuous prompts from base classes to unseen classes?*

To enhance the generalizability of G2P2, we propose an novel alternative G2P2*, which substitute the vanilla prompt tuning with conditional prompt tuning on graphs. G2P2* expands upon G2P2 by integrating a lightweight neural network that


(a) Both G2P2 and G2P2* perform well on the base classes and outperform G2P2+d significantly.

(b) G2P2* shows greater generalizability than G2P2 for the unseen classes.

Fig. 1: From G2P2 to G2P2*: Learning generalizable continuous prompts. Illustrations are based on the Amazon Art dataset (see Sect. VI-A).

generates an input-conditioned prompt token for each node. In contrast to the static prompts of G2P2 that are overly specific to the base classes, the conditional prompts of G2P2* are dynamically responsive to individual nodes, thereby providing greater robustness against class shifts in unseen classes [17]. As illustrated in Fig. 1, G2P2* not only aligns well with the base classes, but also sustains a robust performance on the unseen classes, thereby showcasing its enhanced generalization capabilities. Meanwhile, we note that handcrafted discrete prompts (G2P2+d) show a reasonable level of generalizability. However, they demand extensive manual work and their overall performance on both base and unseen classes is comparatively modest.

Note that G2P2* is not exclusively designed for zero-shot scenarios but serves as an enhancement to G2P2, aiming to improve its handling of zero-shot conditions while preserving its performance in few-shot situations. Essentially, our objective is to maintain strong performance across both base and new class scenarios.

Contributions. We make the following contributions. (1) This is the first attempt to pre-train text and graph encoders jointly for low-resource text classification. (2) We propose a novel model called Graph-Grounded Pre-training and Prompting (G2P2) with three graph interaction-based constrastive strategies in pre-training, as well as discrete and continuous prompts for downstream zero- and few-shot classification, respectively. (3) We extend beyond G2P2 towards handling unseen classes, and propose G2P2*, a conditional prompt tuning approach to generalize the continuous prompts to wider unseen classes while leveraging the base classes. (4) We conduct extensive experiments on four real-world datasets to demonstrate the strength of G2P2 in zero- and few-shot text classification, and the ability of G2P2* to generalize to unseen classes.

A preliminary version of this paper has been published in SIGIR'23 [18]. We provide a summary of the major additions in this version. (1) **Problem**: In Sect. I, we introduced a new problem setting in which zero-shot inference on unseen classes is performed while leveraging a set of base classes. Furthermore, we restructured the introduction to emphasize the motivation, challenges and insights of generalizing prompt tuning to broader unseen classes. (2) **Methodology**: To tackle the limited generalizability of G2P2, we extended G2P2 and introduced a novel conditional prompt tuning approach, named G2P2*, in Sect. V. The conditional prompt tuning can be achieved by incorporating an additional lightweight neural network trained on the base
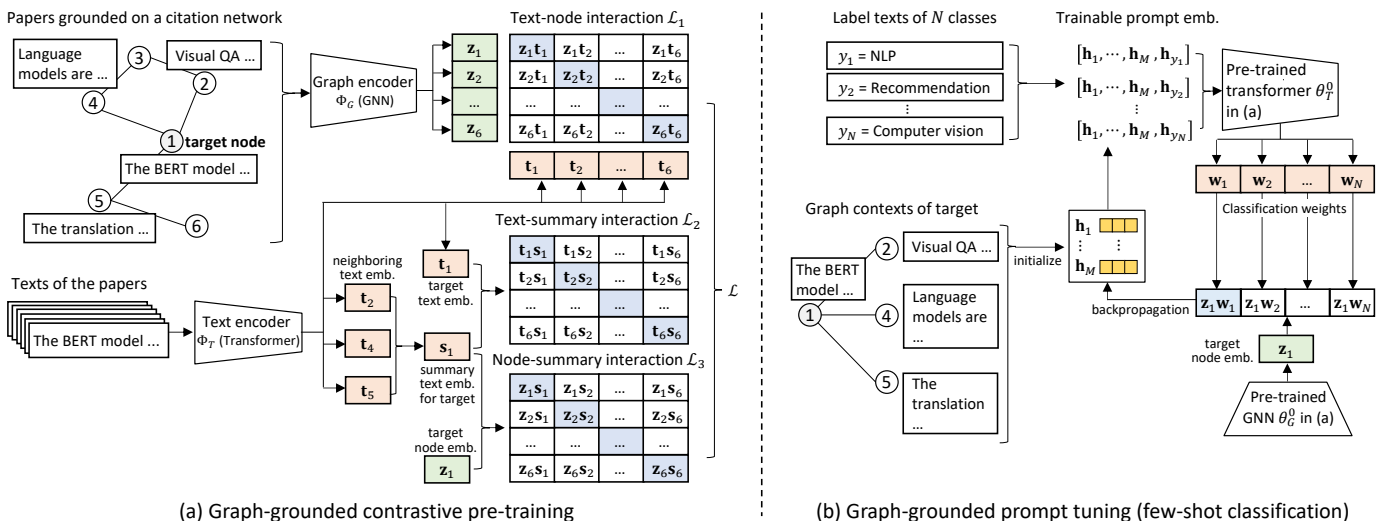
Fig. 2: Overall framework of G2P2. (a) During pre-training, it jointly trains a text and a graph encoder through three contrastive strategies. (b) During testing, it performs prompt-assisted zero- or few-shot classification. Note that part (b) only shows continuous prompt tuning for few-shot classification, while discrete prompts for zero-shot inference and conditional prompt tuning for generalization to wider unseen classes are presented separately in Figs. 3 and 4, respectively.

classes. This network aims to generate an input-dependent token for each input node, which is then fused with learnable global prompt vectors to acquire a customized prompt for each node. (3) **Experiments**: In Sect. VI-D, we conducted additional experiments to evaluate the ability of G2P2* to generalize from base classes to unseen classes. The results demonstrate its excellent generalization capabilities, outperforming G2P2 by 3.5–29.6% for within-domain generalization and 8.8–29.8% for cross-domain generalization.

II. RELATED WORK

Graph neural networks. Inspired by the success of CNN in computer vision, GNNs have emerged to handle non-Euclidean relational data [10], ranging from early semi-supervised models such as GCN [19], GAT [20] and GIN [21], to the more recent self-supervised pre-training paradigm [12]–[14], [22]. Besides their widespread success on graph tasks, they have also been leveraged to improve text-based tasks through knowledge graphs [23] and heterogeneous graphs [24], or multi-modality learning [25]. However, these approaches either employ coarse-grained text treatment, or have decoupled graph and text encoders without fully exploiting the intrinsic relationship between them. Although GLEM [26] integrating both the text and graph structure information with large language models and GNNs, it is not a good low-resource learner.

Language pre-training and prompting. Pre-trained language models [27] have become the most popular backbone in NLP. While earlier PLMs such as GPT [4], BERT [3], XLNet [28] and RoBERTa [29] still have affordable model size, recent introductions such as T5 [30] and GPT-3 [6] produce massive models with billions of parameters. To avoid the high fine-tuning cost on these large models, prompting [31] starts to receive more attention. A prompt is a special template to pad the task input, with a goal of extracting useful knowledge from PLMs to flexibly adapt to downstream tasks. Fueled

by the success of GPT-3, numerous prompting methods including discrete natural language prompt [32]–[34] and continuous prompt [7]–[9], [35], [36] have emerged. The strength of prompting has been validated in a wide range of NLP applications, including text classification [37]–[41], machine translation [42] and relation extraction [43], [44]. More recently, prompting has also been applied to GNNs for graph-centric tasks such as node classification [45], [46] and graph classification [47], [48], but they cannot utilize fine-grained text information.

Zero- or few-shot paradigms. Broadly speaking, our setting is also related to other learning paradigms. For example, in semi-supervised learning [49]–[51], each class may only have a few examples, but all classes must be seen in training and they cannot handle any novel class during testing. Meta-learning [52]–[60] is another popular paradigm that supports few-shot learning. However, large-scale labeled data are still required in a so-called "meta-training" phase, to support the few-shot learning of novel classes during "meta-testing". In contrast, we only need label-free data for pre-training, without requiring any meta-training phase that would consume large-scale labeled data. Separately, there also exists joint consideration of image and text data using a contrastive pre-training strategy for zero- or few-shot classification [17], [61], [62]. In our work, graph data are significantly different from images, which provide various types of interaction between texts. On graphs, zero-shot node classification has also been done [63]. It relies heavily on the availability of Wikipedia pages or other side information to generate class prototype embeddings. However, it is very labor-intensive to find and curate the right side information, especially when there are a large number of classes and/or novel classes emerge frequently.

III. PRELIMINARIES

In this section, we introduce relevant concepts and our low-resource classification settings.

Graph-grounded text corpus. Consider a set of documents $\mathcal{D}$, which is grounded on a graph $\mathcal{G} = (\mathcal{D}, \mathcal{E}, \mathbf{X})$ such that each document $d_i \in \mathcal{D}$ is a node v_i in the graph. The documents are linked via edges in $\mathcal{E}$, which are formed based on the application (*e.g.*, if each document represents an article, the edges could be citations between articles). Each node v_i is also associate with a feature vector $\mathbf{x}_i$, given by the input feature matrix $\mathbf{X}$. Finally, each document/node[1] has a class label (*e.g.*, the topic of the article).

Low-resource classification. A low-resource task consists of a support set $\mathcal{S}$ and a query set $\mathcal{Q}$. The support set $\mathcal{S}$ contains N classes, and each class has K labeled examples where K is a small number (*e.g.*, 1 or 5), known as N-way K-shot classification. The query set $\mathcal{Q}$ contains one or more unlabeled instances belonging to the N classes in the support set. Our goal is to classify the instances in the query set based on the labeled examples in the support set. Unlike episodic few-shot meta-learning [52] which has both training tasks and testing tasks, we only have testing tasks; in the training stage, we perform self-supervised pre-training on label-free data only. As a special case, tasks with $K = 0$ are known as zero-shot classification, which means there is no labeled example at all and we can only rely on class metadata (*e.g.*, class label text).

IV. PROPOSED APPROACH: G2P2

In this section, we introduce our novel model G2P2 for low-resource text classification.

A. Overview of G2P2

As shown in Fig. 2, our model G2P2 consists of two stages: (a) graph-grounded constrastive pre-training, and (b) prompt-tuning for low-resource classification.

During pre-training as shown in Fig. 2(a), we learn a dual-modal embedding space by jointly training a text encoder and graph encoder in a self-supervised fashion, since a document also exists as a node on the graph. More specifically, we use a transformer-based text encoder and a GNN-based graph encoder. The transformer takes the text on each node (*i.e.*, document) as the input, and outputs a text embedding vector $\mathbf{t}_i$ for node v_i. On the other hand, the GNN takes the graph and node features as input, and generates a node embedding vector $\mathbf{z}_i$ for node v_i. Subsequently, in the dual-modal embedding space, we align the text and graph representations on the same or related nodes through three contrastive strategies based on different types of interaction on the graph.

In downstream testing, we use handcrafted discrete prompts (Fig. 3) together with the label text for zero-shot classification. For few-shot classification, we use continuous prompts to pad the label text (Fig. 2(b)).

B. Graph-grounded contrastive pre-training

As shown in Fig. 2(a), the graph-grounded pre-training learns a dual-modal embedding space by jointly training a text encoder

---

[1]We will use "node" and "document" interchangeably given their one-one correspondence in our context.

and a graph encoder, based on three types of interaction on the underlying graph.

Dual encoders. The text encoder is a transformer [5], which we denote Φ_T. Given a document d_i, the text encoder outputs the d-dimensional embedding vector of d_i, denoted $\mathbf{t}_i \in \mathbb{R}^d$:

$$\mathbf{t}_i = \Phi_T(d_i; \theta_T), \tag{1}$$

where θ_T represents the parameter set of the transformer. Correspondingly, let $\mathbf{T} \in \mathbb{R}^{|\mathcal{D}| \times d}$ represents the text embedding matrix for all documents.

Meanwhile, a document d_i is also a node v_i in the graph. We choose a classic GNN called graph convolutional network (GCN) [19] as the graph encoder, denoted Φ_Z. It similarly outputs an embedding vector $\mathbf{z}_i \in \mathbb{R}^d$ for a given node v_i:

$$\mathbf{z}_i = \Phi_Z(v_i; \theta_G), \tag{2}$$

where θ_G represents the parameter set of the GCN. Likewise, let $\mathbf{Z} \in \mathbb{R}^{|\mathcal{D}| \times d}$ represents the graph embedding matrix for all nodes.

Text-node interaction. Our graph-grounded texts naturally implies a bijection between nodes and texts, where each document d_i corresponds to the node v_i in the graph. Inspired by the pairing of image and its caption text [61] and the mapping of content and node sequences [64], we design a pre-training strategy to predict which text document matches which node in the graph.

Specifically, given n documents and the corresponding n nodes, there are n^2 possible document-node pairs $\{(d_i, v_j) \mid i, j = 1, \ldots, n\}$. Among them, only n pairs with $i = j$ are true matching, whereas the remaining $n^2 - n$ pairs are false matching. As our first contrastive strategy, we exploit the bijective interaction between texts and nodes on the graph, to maximize the cosine similarity of the n matching pairs, while minimizing the cosine similarity of the $n^2 - n$ unmatching pairs. To compute the cosine similarity for the n^2 pairs, we first perform a row-wise L2 normalization on embedding matrices $\mathbf{T}$ and $\mathbf{Z}$ to obtain $\tilde{\mathbf{T}}$ and $\tilde{\mathbf{Z}}$, respectively. We then compute a node-text similarity matrix $\mathbf{\Lambda}_1 \in \mathbb{R}^{n \times n}$ to capture the pairwise cosine similarity, as follows.

$$\mathbf{\Lambda}_1 = \left(\tilde{\mathbf{Z}} \tilde{\mathbf{T}}^\top \right) \cdot \exp(\tau), \tag{3}$$

where $\tau \in \mathbb{R}$ is a trainable temperature parameter to scale the similarity values [61].

REMARK. Although $\mathbf{\Lambda}_1 \in \mathbb{R}^{n \times n}$ is a dense matrix, it is constructed in a batch-wise manner for practical implementation. That is, n is not the total number of documents but the relatively small batch size, and thus the overhead is negligible. $\mathbf{\Lambda}_2$ and $\mathbf{\Lambda}_3$ will be introduced later following the same treatment.

To formulate the contrastive loss based on the text-node bijective interaction, we adapt the *multi-class N-pair loss* [65], [66], by considering both the row-wise and column-wise cross entropy loss w.r.t. the row or column index. For instance, the i-th row of $\mathbf{\Lambda}_1$ represents the similarity scores between node v_i and every document, in which the row index i indicates the ground truth document d_i that matches v_i.

$$\mathcal{L}_1 = \tfrac{1}{2} \left(\mathrm{CE}(\mathbf{\Lambda}_1, \mathbf{y}) + \mathrm{CE}(\mathbf{\Lambda}_1^\top, \mathbf{y}) \right), \tag{4}$$

where $\mathbf{y} = (1, 2, \ldots, n)^\top$ is the label vector for contrastive training, and CE denotes the cross entropy loss applied to the input matrix $\mathbf{\Lambda}_1$ or $\mathbf{\Lambda}_1^\top$ in a row-wise manner.

Text-summary interaction. Apart from the bijective text-node interaction, we further exploit higher-order interactions on the graph. In particular, each document has a set of neighboring documents defined by graph topology. The neighboring documents can be understood as a summary of the target document given the semantic relatedness between them. For example, on a e-commerce network, the products purchased by a user naturally portray a summary of the user and vice versa. Without loss of generality, we employ a simple mean pooling to generate the summary embedding $\mathbf{s}_i \in \mathbb{R}^d$ as follows.

$$\mathbf{s}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{t}_j. \qquad (5)$$

For efficiency, we only sample a fixed number of neighboring documents to generate the summary. Then, let $\mathbf{S} \in \mathbb{R}^{n \times d}$ denote the summary text embedding matrix for all documents.

Hence, as our second contrastive strategy, we seek to align the text embedding of each document and its corresponding summary text embedding, based on the text-summary interaction derived from graph neighborhood. In other words, we maximize the cosine similarity of the n matching pairs of document and its neighborhood-based summary, while minimizing the cosine similarity of the $n^2 - n$ unmatching pairs. Specifically, we first follow Eq. (3) to construct a text-summary similarity matrix $\mathbf{\Lambda}_2 \in \mathbb{R}^{n \times n}$:

$$\mathbf{\Lambda}_2 = \left(\tilde{\mathbf{T}} \tilde{\mathbf{S}}^\top \right) \cdot \exp(\tau). \qquad (6)$$

Subsequently, we apply the same contrastive loss following Eq. (4), as follows.

$$\mathcal{L}_2 = \frac{1}{2} \left(\text{CE}(\mathbf{\Lambda}_2, \mathbf{y}) + \text{CE}(\mathbf{\Lambda}_2^\top, \mathbf{y}) \right), \qquad (7)$$

Node-summary interaction. The neighborhood-based summary for document d_i also serves as a semantic description of node v_i. Mirroring the text-summary interaction, as our third contrastive strategy, we seek to align the node embedding and its neighborhood-based summary text embedding. We similarly compute a node-summary similarity matrix $\mathbf{\Lambda}_3 \in \mathbb{R}^{n \times n}$, and formulate the corresponding contrastive loss $\mathcal{L}_3$.

$$\mathbf{\Lambda}_3 = \left(\tilde{\mathbf{Z}} \tilde{\mathbf{S}}^\top \right) \cdot \exp(\tau), \qquad (8)$$

$$\mathcal{L}_3 = \frac{1}{2} \left(\text{CE}(\mathbf{\Lambda}_3, \mathbf{y}) + \text{CE}(\mathbf{\Lambda}_3^\top, \mathbf{y}) \right). \qquad (9)$$

Overall pre-training objective. Finally, we integrate the three contrastive losses based on the text-node, text-summary and node-summary interactions. We obtain a pre-trained model $\theta^0 = (\theta_T^0, \theta_G^0)$ consisting of the parameters of the dual encoders, given by

$$\theta^0 = \arg \min_{\theta_T, \theta_G} \mathcal{L}_1 + \lambda(\mathcal{L}_2 + \mathcal{L}_3), \qquad (10)$$

where $\lambda \in \mathbb{R}^+$ is a hyperparameter to balance the contribution from summary-based interactions.

The pre-training procedure is outlined in Algorithm 1, which has the following complexity per epoch. Let $|\mathcal{D}|$ be the number of documents, η be the number of neighbors sampled to

---

Algorithm 1 PRE-TRAINING PROCEDURE OF G2P2

Require: A graph-grounded text corpus $\mathcal{G} = (\mathcal{D}, \mathcal{E}, \mathbf{X})$.
Ensure: Pre-trained weights of text encoder θ_T^0, graph encoder θ_G^0.
1: $\theta_T^0, \theta_G^0 \leftarrow$ parameters initialization;
2: **while** not converged **do**
3:     sample batches of documents from $\mathcal{D}$;
4:     **for** each batch **do**
5:         **for** each node v_i/document d_i in the batch **do**
6:             compute d_i's text embedding $\mathbf{t}_i$;   ▷ Eq. (1)
7:             compute v_i's node embedding $\mathbf{z}_i$;   ▷ Eq. (2)
8:             compute v_i's summary embedding $\mathbf{s}_i$;   ▷ Eq. (5)
9:         **end for**
10:       compute similarity matrices $\mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \mathbf{\Lambda}_3$;   ▷ Eqs. (3), (6), (8)
11:       compute contrastive losses $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$;   ▷ Eqs. (4), (7), (9)
12:       update the overall loss $\mathcal{L}$;   ▷ Eq. (10)
13:       $\theta_T^0, \theta_G^0 \leftarrow$ update via backpropagation
14:     **end for**
15: **end while**
16: **return** θ_T^0, θ_G^0.

---

generate the summary embedding in Eq. (5), and β be the batch size. First, the cost of generating the three types of embeddings (lines 5–8) per epoch is $O(|\mathcal{D}|\eta)$, given that calculating the summary embedding needs go through η neighbors. Second, the cost of calculating the three similarity matrices in each batch is $O(\beta^2)$, and the total cost per epoch is $O\left(\frac{|\mathcal{D}|}{\beta}\beta^2\right) = O(|\mathcal{D}|\beta)$ given $\frac{|\mathcal{D}|}{\beta}$ batches in an epoch. Thus, the overall complexity is $O(|\mathcal{D}|(\eta + \beta))$, which is linear in the number of documents given that η and β are small constants. In our implementation, we set $\eta = 3$ and $\beta = 64$.

C. Prompting joint graph-text model

After pre-training our graph-text model, it is non-trivial to apply it to low-resource classification. To narrow the gap between pre-training and downstream tasks, the traditional "pre-train, fine-tune" paradigm typically introduces a new projection head for the downstream task, which will be fine-tuned together with the whole pre-trained model. However, under the low-resource setting, it is neither effective nor efficient to update the entire model with a huge number of parameters. Without updating massive PLMs, prompting has recently emerged as a powerful alternative to fine-tuning in NLP [31], although prompting has not been explored for graph-text models with jointly pre-trained structural and textual information.

In this part, we elaborate zero-shot classification using handcrafted discrete prompts, as the absence of labeled data in zero-shot tasks cannot support directly learnable prompts. We further discuss automated continuous prompt tuning for few-shot classification.

Discrete prompts for zero-shot classification. In N-way zero-shot classification, out of N classes, we predict the class which has the highest similarity to the given node. As illustrated by the diagram in Fig. 3, the classification weights can be generated by the text encoder based on the class label texts [67], without requiring any labeled sample for the classification task. Specifically, the weight vector $\mathbf{w}_y$ for class $y \in \{1, 2, \ldots, N\}$ is the output from the pre-trained text encoder, *i.e.*,

$$\mathbf{w}_y = \phi_T(\text{"prompt [CLASS]"}; \theta_T^0). \qquad (11)$$

Here "prompt [CLASS]" is a prompt template, where [CLASS] refers to the label text of the target class y (*e.g.*, "NLP" for paper
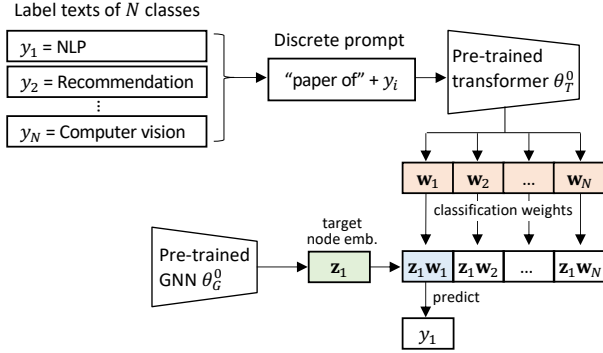
Fig. 3: Schematic diagram for zero-shot classification. The pre-trained models θ_G^0 and θ_T^0 are obtained from Fig. 2(a).

area classification), and `prompt` is a handcrafted sequence of natural language tokens to signal the relevance of the label text (*e.g.*, "`paper of NLP`" helps focus on the topic of the paper). In the simplest case, `prompt` can be an empty string so that we only rely on the label text. Note that discrete tokens are still converted to continuous word embeddings as input to the text encoder; for brevity we omit this step in Eq. (11).

Then, the class distribution given node representation $\mathbf{z}_i$ is predicted as

$$p(y \mid \mathbf{z}_i) = \frac{\exp\left(\langle \mathbf{z}_i, \mathbf{w}_y \rangle\right)}{\sum_{y=1}^N \exp\left(\langle \mathbf{z}_i, \mathbf{w}_y \rangle\right)}, \quad (12)$$

where $\langle \cdot, \cdot \rangle$ is cosine similarity.

Continuous prompts for few-shot classification. The problem with discrete prompts is that they are difficult to optimize, given that PLMs are intrinsically continuous. Substituting discrete natural language prompts with learnable continuous prompts, prompt tuning [8], [9], [68] can automate the optimization of prompts when some labeled data is available. Hence, in the few-shot setting, we explore prompt tuning to cue in the relevant structural and semantic information from our jointly pre-trained graph-text model.

Specifically, instead of a sequence of discrete tokens, we take a sequence of continuous embeddings $[\mathbf{h}_1, \cdots, \mathbf{h}_M, \mathbf{h}_{\text{CLASS}}]$ as the prompt, where M is a hyperparameter indicating the number of prompt tokens, each $\mathbf{h}_m$ ($m \leq M$) is a trainable vector, and $\mathbf{h}_{\text{CLASS}}$ is the word embedding sequence of the target class label. The continuous prompt is fed as input to the text encoder to generate the classification weights for each class y:

$$\mathbf{w}_y = \phi_T([\mathbf{h}_1, \cdots, \mathbf{h}_M, \mathbf{h}_{\text{CLASS}}]; \theta_T^0), \quad (13)$$

where each $\mathbf{h}_m$ ($m \leq M$) has the same dimension as the input word embeddings to the text encoder.

Using the same softmax layer in Eq. (12), we further update the continuous prompt embeddings using the labeled support set of the few-shot task by minimizing a cross entropy loss, whilst freezing the parameters of the dual encoders. This prompt tuning process is both data- and computation-efficient, given the small number of learnable parameters in the prompt.

Furthermore, existing prompt tuning methods either initialize the prompt embeddings randomly [8], [68] or using the word embeddings of handcrafted discrete prompts [62]. While random initialization is non-informative and more prone to

local optimum, it is still difficult to pick the right discrete prompts for initialization. Therefore, we take the advantage of graph structures to initialize the prompt embeddings.

Specifically, given a node v_i, we define its *graph contexts* as its neighbor set $\{v_j \mid j \in \mathcal{N}_i\}$. Due to the underlying semantic relatedness, the graph contexts of the few-shot examples carry strong signals about the task, which can be exploited to improve the initialization. For each document/node v_i in the task support set, we sample η nodes from its graph contexts. For v_i itself and each context node sampled, we truncate its corresponding document to M words, and convert it to a sequence of M word embedding vectors, each having the same dimension as the vector $\mathbf{h}_m$ ($m \leq M$) in our continuous prompt. Hence, for each support node, we would obtain $\eta + 1$ such sequences; in an N-way K-shot task, there is a total of $NK(\eta + 1)$ sequences. We take the average of these embedding sequences to initialize the learnable prompt vectors $\mathbf{h}_1, \ldots, \mathbf{h}_M$, which is derived from graph contexts and thus could provide a more informative starting point than random initialization.

V. CONDITIONAL PROMPT TUNING: G2P2*

In this section, we first review the limitations of G2P2 in generalizing to unseen classes, and propose G2P2* to overcome the limitation based on conditional prompt tuning.

A. Limitation of G2P2

The proposed continuous prompt tuning in Sect. IV-C aims to learn a collection of trainable vectors in each few-shot task. Compared to handcrafted discrete prompts, prompt tuning is automated and tends to be more robust than manual engineering. However, it still requires some labeled data, and thus has limited capability in dealing with zero-shot inference, in which the test set involves the so-called "unseen" classes without any labeled instance. Due to class shift, directly using prompts learned from a set of "base" classes with labeled data do not generalize well to broader unseen classes without labeled data.

Specifically, consider a subset of classes in each dataset as *base* classes and the rest as *unseen* classes. In a base class, only a small number of labeled instances are available for selecting discrete prompts or tuning continuous prompts, reserving the majority for testing. In contrast, all instances of the *unseen* classes are used solely for testing, without revealing any labeled instance for prompt selection or tuning. In this setting, we are essentially conducting zero-shot inferences on the unseen classes, using the prompts learned from the base classes.

As illustrated in Fig. 1, we can observe a critical limitation in G2P2. On one hand, in Fig. 1(a), G2P2 can learn effective prompts to accurately identify base classes, such as "ink" and "pencils". On the other hand, in Fig. 1(b), when using the same prompts optimized for the base classes, the accuracy of G2P2 markedly diminishes when facing novel, unseen classes like "oil paint" and "canvas". The significant decrease in performance occurs despite the consistent nature of the task, which involves the classification of item categories in the same Art domain. One potential reason is that the prompts learned by G2P2 overfit to the base classes, resulting in poor generalization to unseen classes. The issue originates from the
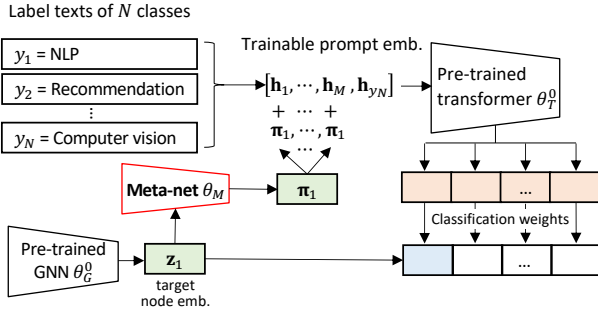
Fig. 4: Schematic diagram for conditional prompt tuning in G2P2*. The pre-trained models θ_G^0 and θ_T^0 are obtained from Fig. 2(a). The classes are base class during tuning, and unseen classes during zero-shot inference.

static prompt design, where the prompts, once learned, are tailored specifically to the base classes. Meanwhile, G2P2+d with handcrafted discrete prompts demonstrates a comparatively higher level of generalization with only a small performance decrease on the unseen classes. However, handcrafted prompts require significant manual labor and can give inconsistent performance on different tasks or datasets.

B. Proposed extension: G2P2*

To generalize prompt tuning to wider unseen classes, we explore conditional prompt tuning [17]. The core principle is to *condition* a prompt on each individual input instance (*i.e.*, each document/node in our context), rather than learning a static prompt tailed to a specific task or a fixed set of base classes. At the same time, to ensure parameter efficiency, we extend G2P2 by adding a lean neural network to generate an input-conditional token (vector) for each node, which is then integrated with the learnable prompt vectors. We refer the conditional variant of our method as G2P2*. Intuitively, the conditional token is analogous to node captioning or document summarization, which is equivalent to a description for each node or document. Hence, conditional prompts are more generalizable: They are optimized to depict each instance and thus more resilient to class shift, rather than being confined to certain specific classes.

Recall that in the unconditional version, we learn M global prompt tokens that can be used with all nodes in a task. In contrast, in the conditional version, we need M prompt tokens that are specific to each input node. A straightforward way is to increase the parameters, whereby we associate each node with M learnable vectors directly, or train M neural networks to generate M distinct vectors for each node. In either design, the model size is substantially larger than the global prompt vectors in the original G2P2. Inspired by the Meta-net architecture [17], [69], we advocate for a more parameter-efficient design that has demonstrated impressive results in low-resource settings. Instead, we introduce a light-weight neural network, known as a Meta-net, on top of the M global prompt vectors to generates one conditional token (vector) for each input node, which is subsequently fused with the global prompt vectors.

A schematic diagram of the conditional prompt tuning in G2P2* is sketched in Fig. 4. Specifically, our Meta-net Φ_M employs a dual-layer multi-layer perceptron (MLP) with a

bottleneck structure [70]. Given a node v_i with its embedding $\mathbf{z}_i$, the Meta-net generates a conditional token π_i, *i.e.*,

$$\pi_i = \Phi_M(\mathbf{z}_i; \theta_M). \quad (14)$$

Then, the m-th prompt vector for node v_i, denoted $\mathbf{h}_{m,i}$, is obtained by

$$\mathbf{h}_{m,i} = \mathbf{h}_m + \pi_i, \quad (15)$$

where $\mathbf{h}_m$ is the m-th global prompt vector in the original G2P2. Note that the conditional token π_i should have the same dimension as the global prompt vector $\mathbf{h}_m$. The conditional prompt for node v_i is thus $[\mathbf{h}_{1,i}, \cdots, \mathbf{h}_{M,i}, \mathbf{h}_{\text{CLASS}}]$, where $\mathbf{h}_{\text{CLASS}}$ is the word embedding sequence of the target class label identical to that used in the continuous prompt. Subsequently, the conditional prompt is fed as input to the text encoder to output the classification weights, which means the weights are also conditioned on the input node. Specifically, the classification weight for node v_i and class y is

$$\mathbf{w}_{y,i} = \phi_T([\mathbf{h}_{1,i}, \cdots, \mathbf{h}_{M,i}), \mathbf{h}_{\text{CLASS}}]; \theta_T^0). \quad (16)$$

Thus, the prediction probability is computed as

$$p(y \mid \mathbf{z}_i) = \frac{\exp\left(\langle \mathbf{z}_i, \mathbf{w}_{y,i} \rangle\right)}{\sum_{y=1}^N \exp\left(\langle \mathbf{z}_i, \mathbf{w}_{y,i} \rangle\right)}. \quad (17)$$

The training process is similar to G2P2's prompt tuning: we train the M global prompt vectors and Meta-net by minimizing cross-entropy through few-shot node classification on the base classes. Meta-net is just another way of prompt learning. It extends the single global prompt to node-specific prompts. Besides, meta-net is small, it is similar to other Parameter Efficient Fine-Tuning methods [71] that also use a small network to work with Large models.

VI. EXPERIMENTS

We conduct extensive experiments to evaluate G2P2 and G2P2*, with comparison to state-of-the-art baselines and detailed model analyses.

A. Experimental setup

Datasets. Four public graph-grounded text corpora are used, as summarized in Tab. 1. The first dataset is **Cora** [1]: known as the "Cora Research Paper Classification" dataset, it is a collection of research papers that are linked to each other through citations. The abstract of a paper is deemed a text document. The papers are classified into a topic hierarchy with 70 leaves. Note that we are using a more comprehensive version of Cora, which is larger and has more classes than the version used elsewhere [19]. The other three datasets, namely, **Art**, **Industrial** and **Music Instruments (M.I.)** , are all Amazon review collections [72], respectively from three broad areas, namely, arts, crafts and sewing (Art), industrial and scientific (Industrial), and musical instruments (M.I.). Product descriptions and aggregated user reviews are treated as text, with reviews linking users to products and aiding in detailed classification into many specific subcategories. User reviews also enhance understanding of product-related text. For all

Table 1: Statistics of datasets.

Dataset	Cora	Art	Industrial	M.I.
# Documents	25,120	1,615,902	1,260,053	905,453
# Links	182,280	4,898,218	3,101,670	2,692,734
# Avg. doc length	141.26	54.23	52.15	84.66
# Avg. node deg	7.26	3.03	2.46	2.97
# Total classes	70	3,347	2,462	1,191

datasets, we employ the word2vec algorithm [16] to obtain the 128-dimensional word embeddings of each word in the text documents. Then, for each node, we average the word embedding vectors of all the words in its document, and the averaged vector is used as the node's input features for the GNN-based methods.

Task construction. We perform zero- or few-shot text classification. We adopt a *5-way* setting, *i.e.*, we sample five classes from all the classes to construct a task. In each task, we construct a K-shot support set by further sampling K examples from each class for $K \in \{0, 1, \ldots, 5\}$, and a validation set of the same size as the support set. The remaining examples form the query set. Note that the support set is labeled and serve as the task training data, whereas the query set is unlabeled and used for evaluation. Note that in our experiment all the classes are used—it is only that each task involves 5 classes, and we have multiple tasks during testing to cover all the classes. This is a typical task setup [52], allowing for a comprehensive evaluation under different class combinations. The reported results are averaged over all the tasks on each dataset.

Baselines for few-shot classification. We consider competitive baselines from four categories.

(1) *End-to-end GNNs*, which are graph neural networks trained in a supervised, end-to-end manner from random initialization, including: **GCN** [19]; **SAGE$_{sup}$**, the supervised version of GraphSAGE [73]; **TextGCN** [74], a GCN-based model on a text graph constructed from word co-occurrence and document-word relations, which jointly learns the embeddings of both words and documents.

(2) *Pre-trained/self-supervised GNNs*, which are pre-trained using pretext tasks without labeled data, followed by fine-tuning or fitting a classification head while freezing the model parameters, including: **GPT-GNN** [14], a GNN pre-training approach by a self-supervised graph generation task; **DGI** [12], a GNN pre-training approach that maximizes the mutual information between local- and global-level representations; **SAGE$_{self}$** [73], the self-supervised version of GraphSAGE, encouraging similar embeddings for neighboring nodes and distinct embeddings for non-adjacent nodes. Both DGI and SAGE$_{self}$ freeze the pre-trained weights, and fit a logistic regression model for the downstream classification.

(3) *Pre-trained transformers*, which are pre-trained using masked language modeling, and then are fine-tuned together with a randomly initialized classification head (*e.g.*, a fully connected layer) for the downstream classification, including: **BERT** [3]; **RoBERTa** [29]; **BERT***** and **RoBERTa*****, variants of BERT and RoBERTa, which are obtained by further pre-training the pre-trained BERT and RoBERTa, respectively, using masked language modeling on our datasets, to mitigate the domain gap between our datasets and the corpus used for

pre-training BERT and RoBERTa.

(4) *Prompt tuning*: **P-Tuning v2** [68], is a version of prefix-tuning [7] optimized and adapted for natural language. It uses deep prompt tuning, which applies continuous prompts for every layer of the pre-trained language model.

Note that our setting is distinct from few-shot learning under the meta-learning paradigm [52], as there is no few-shot tasks for the meta-training phase. Hence, we cannot use state-of-the-art meta-learning models for comparison. Besides, two of the baselines we compared, DGI and SAGE$_{self}$, have adopted a form of linear probe which is known to be a strong few-shot learner [75].

Baselines for zero-shot classification. We only compare with decoder-based LLMs and encoder-based PLMs, as all the other methods require at least one shot to work. Firstly, for LLMs with fewer than 10 billion parameters, we used Qwen-7B [76], BLOOMz-7B [77], and the most capable openly available LLM to date, Llama-3-8B [78]. For models with more than 10 billion parameters, we utilized Baichuan-13B [79], which is currently the open-source model with the most training data at the 13 billion parameter scale. And we used Vicuna-13B [80], which matches over 90% of the quality of OpenAI's ChatGPT [81] and Google Bard [82] while outperforming models like LLaMA and Stanford Alpaca [83] in over 90% of cases. Secondly, for each encoder-based PLM, we use the discrete prompt [CLASS] (*i.e.*, the label text alone). We also evaluate handcrafted prompts "prompt [CLASS]", where prompt is a sequence of tokens found by prompt engineering, and annotate the model name with "+d". We compute the similarity between the target document and the label text of each class (with or without prompt), and predict the most similar class following Fig. 3.

Parameter settings. For G2P2, the text encoder is a transformer [5]. Following CLIP [61], we use a 63M-parameter, 12-layer 512-wide model with 8 attention heads. It operates on a lower-cased byte pair encoding (BPE) representation of the texts with a 49,152 vocabulary size [84]. The max sequence length is capped at 128. The graph encoder employs a GCN [19], using two layers [73] with a LeakyReLU activation, each with 128 dimensions [85]. The pre-training of our model starts from scratch without initializing the graph and text encoders with previously pre-trained weights. λ in Eq. (10) is set to 0.1 on Cora, and set to 10 on the three amazon review datasets, which were chosen from $\{0.01, 0.1, 1, 10, 100\}$ according to the accuracy on validation data. The number of learnable prompt tokens, M in Eq. (13), is set to 4, which was chosen from $\{2, 4, 8, 16, 32\}$ according to the accuracy on validation data. We use the Adam optimizer with the learning rate 2×10^{-5} with 2 training epochs, and a batch size of 64 in pre-training, referring to Hugging Face's [86] example settings. The text embedding size is 128, same to the output from the graph encoder. To generate the summary embedding and the context-based prompt initialization, the number of neighboring nodes sampled is 3. For prompt tuning, we set the learning rate as 0.01, which was chosen from $\{0.0001, 0.001, 0.01, 0.1\}$ according to the accuracy on validation data. Lastly, the Meta-net of G2P2*** is built with a two-layer bottleneck structure (Linear-ReLU-Linear), with the hidden-layer dimension being 8.

Table 2: *Five-shot* classification performance (percent) with 95% confidence intervals.

In each column, the best result among all methods is **bolded** and the best among the baselines is <u>underlined</u>. Improvement by G2P2 is calculated relative to the best baseline. * indicates that our model significantly outperforms the best baseline based on the two-tailed t-test ($p < 0.05$).

	Cora		Art		Industrial		M.I.	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
GCN	41.15±2.41	34.50±2.23	22.47±1.78	15.45±1.14	21.08±0.45	15.23±0.29	22.54±0.82	16.26±0.72
SAGE$_{sup}$	41.42±2.90	35.14±2.14	22.60±0.56	16.01±0.28	20.74±0.91	15.31±0.37	22.14±0.80	16.69±0.62
TextGCN	59.78±1.88	55.85±1.50	43.47±1.02	32.20±1.30	53.60±0.70	45.97±0.49	46.26±0.91	38.75±0.78
GPT-GNN	76.72±2.02	72.23±1.17	65.15±1.37	52.79±0.83	62.13±0.65	54.47±0.67	67.97±2.49	59.89±2.51
DGI	<u>78.42</u>±1.39	<u>74.58</u>±1.24	65.41±0.86	53.57±0.75	52.29±0.66	45.26±0.51	68.06±0.73	60.64±0.61
SAGE$_{self}$	77.59±1.71	73.47±1.53	76.13±0.94	65.25±0.31	71.87±0.61	65.09±0.47	<u>77.70</u>±0.48	<u>70.87</u>±0.59
BERT	37.86±5.31	32.78±5.01	46.39±1.05	37.07± 0.68	54.00±0.20	47.57±0.50	50.14±0.68	42.96±1.02
BERT*	27.22±1.22	23.34±1.11	45.31±0.96	36.28±0.71	49.60±0.27	43.36±0.27	40.19±0.74	33.69±0.72
RoBERTa	62.10±2.77	57.21±2.51	72.95±1.75	62.25±1.33	76.35±0.65	70.49±0.59	70.67±0.87	63.50±1.11
RoBERTa*	67.42±4.35	62.72±3.02	74.47±1.00	63.35±1.09	77.08±1.02	71.44±0.87	74.61±1.08	67.78±0.95
P-Tuning v2	71.00±2.03	66.76±1.95	<u>76.86</u>±0.59	<u>66.89</u>±1.14	<u>79.65</u>±0.38	<u>74.33</u>±0.37	72.08±0.51	65.44±0.63
G2P2-p	79.16±1.23	74.99±1.35	79.59±0.31	68.26±0.43	80.86±0.40	74.44±0.29	81.26±0.36	74.82±0.45
G2P2	**80.08***±1.33	**75.91***±1.39	**81.03***±0.43	**69.86***±0.67	**82.46***±0.29	**76.36***±0.25	**82.77***±0.32	**76.48***±0.52
(improv.)	(+2.12%)	(+1.78%)	(+5.43%)	(+4.44%)	(+3.53%)	(+2.7%)	(+6.53%)	(+7.92%)

B. Performance of G2P2

We evaluate the performance under various few-shot settings.

Five-shot setting. In Tab. 2, we first compare the performance of G2P2 with baselines under the *5-shot* setting. G2P2 emerges as the winner consistently, outperforming the best baseline by around 2–8% with statistical significance.

We also make a few more observations. Firstly, among the GNNs, pre-trained/self-supervised models tend to perform better than the end-to-end approaches, since the latter heavily rely on labeled data. Among the former, DGI and SAGE$_{self}$ perform better as they are a form of linear probe, known to be a strong few-shot learner [75]. Note that, instead of using word2vec embeddings [16] of raw texts as node features, we also tired using the pre-trained RoBERTa [29] to generate the node features for DGI and SAGE$_{self}$. However, doing so does not bring any improvement, showing that it is ineffective to simply combine a language model and GNN in a decoupled manner. In contrast, our proposed model jointly learns the text and graph encoders through three graph-grounded contrastive strategies. Secondly, PLMs are generally superior to GNNs, illustrating the importance of leveraging texts in a fine-grained way. Additionally, RoBERTa outperforms BERT owing to an improved pre-training procedure [29]. However, further training PLMs on our texts gives mixed results: RoBERTa* slightly outperforms RoBERTa but BERT* is much worse than BERT. That means it is not straightforward to mitigate the domain gap by simply continuing training on the domain texts. Thirdly, the continuous prompt approach P-Tuning v2 achieves competitive results compared to fine-tuning, while having the advantage of being much cheaper than fine-tuning. Nevertheless, it is still significantly outperformed by our model G2P2. Furthermore, G2P2-p without prompt tuning is inferior to G2P2, showing the benefit of continuous prompts.

Settings with fewer shots. In addition to the 5-shot setting, in Fig. 5 we also study the impact of fewer shots on G2P2 and several representative baselines. G2P2 generally performs the best across different shots. In general, the performances of all approaches degrade as fewer shots become available. However,
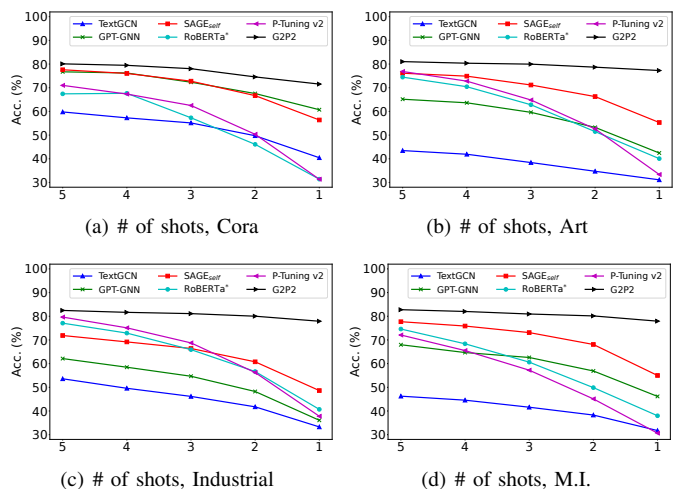

(a) # of shots, Cora

(b) # of shots, Art

(c) # of shots, Industrial

(d) # of shots, M.I.

Fig. 5: Classification performance on different shots.

the baselines suffer significantly under extreme low-resource (*e.g.*, 1- or 2-shot) settings. In contrast, G2P2 remains robust, reporting a relatively small decrease in performance even with just 1 or 2 shots.

It demonstrates the practical value of our proposed model especially when labeled data are difficult or costly to obtain in time. On the other hand, traditional approaches constantly face the challenge of the inability to keep up with the rapid growth of emerging classes in dynamic and open environments [63]. For example, labeling a large volume of texts for novel topics in online articles, or new product categories in open-ended e-commerce platforms, can suffer a substantial time lag.

Zero-shot setting inference. Finally, we report the zero-shot performance in Tab. 3, where our models G2P2 and G2P2+d significantly outperforms both the decoder-based LLMs and encoder-based PLMs baselines. The results particularly demonstrate the effectiveness of our graph-grounded contrastive pre-training in the absence of labeled data, which is crucial to handling evolving classes without any labeled sample in many real-world scenarios. We make several further observations.

Firstly, we specifically utilized five state-of-the-art decoder-

based LLMs. Empirically, those LLMs requires lots of computational resources and time, *e.g.*, a 13B model requires **more than 32GB** of GPU memory for inference with a **batch size of 1**. And it takes **over two weeks** for a single LLM to complete **one inference run** on the Art test set. In contrast, an encoder-based model like our G2P2 takes less than five hours to complete five inference runs on the largest Art test set. As shown in Tab. 3: (1) while decoder-based LLMs provide useful insights, their computational requirements are significantly higher, making them less efficient for the given task; (2) although decoder-based LLMs are famous for zero-shot scenarios, our G2P2 model outperforms them in these settings, providing more accurate results with lower computational overhead.

Secondly, for results about decoder-based PLMs, we can see that handcrafted discrete prompts (*i.e.*, BERT*+d and G2P2+d) can be superior to using label text only (*i.e.*, BERT* and G2P2), showing the effectiveness of additional prompt tokens.

However, finding the optimal discrete prompts often requires significant engineering work. Specifically, for the three approaches with discrete prompts, namely, RoBERTa*+d, BERT*+d and G2P2+d, we have explored more than 10 handcrafted prompt templates on each dataset, which are typically specific to the dataset and require some domain knowledge to devise. While discrete prompts are generally helpful, their effectiveness varies. For example, on the Cora dataset, while "a `model` of [CLASS]" is the best prompt for RoBERTa*+d, it is a bad choice for G2P2+d. Sometimes, generic prompts like "a [CLASS]" can be the best choice. Note that in Tab. 3, we simply report the performance of the best template for each approach and dataset. Hence, using the label text only is still a reasonably good choice.

Zero-shot case study. To better understand the motivation and strength of G2P2, we conduct an zero-shot case study, as illustrated in Tab. 4. We randomly selected a testing sample from a 5-way task in the Cora test set, and both the original document and label text are presented in the table.

Decoder-based LLMs cannot provide accurate classification results when only given the original document and label text. Therefore, after experimenting with various prompts, we crafted the instructions shown in blue in the table. It's evident that zero-shot classification tasks remain challenging for popular decoder-based LLMs. In this example, four of the five label texts are semantically similar, making them difficult to differentiate. Only BLOOMz provided the correct answer, while the other four models produced various errors. Qwen generated irrelevant content, Llama-3 repeated the question without providing reasons, and Vicuna entered a repetitive loop—a common issue with decoder-based LLMs.

In contrast, our encoder-based models are more accurate and reliable due to directly using embeddings containing fine-grained information and computing cosine similarity. They are also more time- and computationally efficient due to having fewer parameters than decoder-based LLMs. Among these encoder-based models, G2P2 not only selected the correct answer but also showed a clear distinction between the similarities of the correct and incorrect answers. This is because using GNNs to encode graph structures expands the usable

Table 3: *Zero-shot* classification accuracy (percent).

	Models	# Params.	Cora	Art	Industrial	M.I.
Decoder	Qwen	7B	21.67	18.65	19.13	19.00
	BLOOMz	7B	48.89	23.60	24.07	20.06
	Llama-3	8B	25.32	18.94	19.70	19.47
	Baichuan	13B	21.67	16.68	19.22	18.68
	Vicuna	13B	16.65	15.36	16.81	15.85
Encoder	RoBERTa	123M	30.46	42.80	42.89	36.40
	RoBERTa*	123M	39.58	34.77	37.78	32.17
	RoBERTa*+d	123M	45.53	36.11	39.40	37.65
	BERT	110M	23.58	35.88	37.32	37.42
	BERT*	110M	23.38	54.27	56.02	50.19
	BERT*+d	110M	26.65	56.61	55.93	52.13
	G2P2	66M	64.75	76.62	76.43	74.44
	G2P2+d	66M	**66.43***	**76.95***	**77.31***	**75.94***
	(improv.)	-	(+35.88%)	(+35.93%)	(+38.00%)	(+45.67%)

(a) Interaction coefficient, λ (b) Prompt length, M
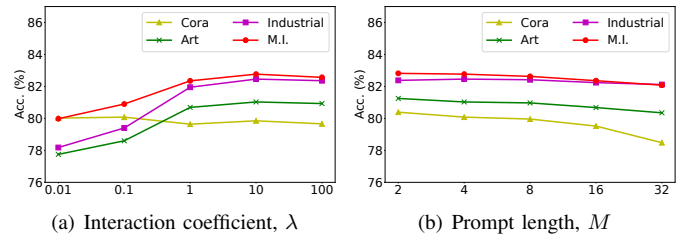
Fig. 6: Hyperparameter study for G2P2.

information from just text to both text and graph structures. Our model effectively leverages these dual modalities to tackle the zero-shot text classification challenge.

C. Model analyses of G2P2

We conduct more in-depth studies on G2P2. By default, we report the classification *accuracy* under the *5-shot* setting.

Ablation study. We first evaluate the contribution from each of the three interaction-based contrastive strategies, by employing different combinations of the proposed loss terms $\mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{L}_3$. As shown in Tab. 5, strategies without $\mathcal{L}_1$ have performed quite poorly, demonstrating that the bijective text-node interaction is the fundamental component of our pre-training. That being said, when further adding $\mathcal{L}_2$ or $\mathcal{L}_3$ to $\mathcal{L}_1$, we still observe a noticeable performance improvement, showing the benefit of incorporating additional graph-based interactions for text data. Lastly, G2P2 with all three loss terms outperforms all 1- or 2-combinations of the losses, demonstrating that the three contrastive strategies are all useful and they are well integrated. Overall, the results reveal that graph information is vital to low-resource text classification, since graph structures reveal rich relationships between documents.

Next, we evaluate the contribution from our prompt-tuning. Specifically, we compare G2P2 with two ablated variants: using label text only without trainable prompt vectors, and randomly initializing the prompt vectors. As reported in Tab. 5, only using label text clearly degrades the classification performance, implying the importance of learning continuous prompts via prompt tuning. Furthermore, our approach G2P2 with context-based initialization for prompt vectors shows a small but consistent advantage over random initialization, implying the usefulness of considering graph structures in prompt tuning.

Table 4: *Zero-shot* case study. An example from Cora, with the ground-truth answer **B**. For encoder-based PLMs, the cosine similarity here is softmax normalised. *Italic* text is the instruction crafted by us for decoder-based LLMs.

Document	this paper develops a new and natural parallel vector model, and shows that for all k 1, the languages recognizable in o(log k n) time and polynomial work in the model are exactly those in nc k . some improvements to other simulations in the literature of parallel models and reversal complexity are given.
Label text	(A) data structures algorithms and theory, computational geometry (B) data structures algorithms and theory, computational complexity (C) data structures algorithms and theory, parallel (D) data structures algorithms and theory, logic (E) artificial intelligence, nlp
Decoder-based LLMs	
Prompt for decoder-based LLMs	*This is a paper abstraction:* this paper develops a new and natural parallel vector model, and shows that for all k 1, the languages recognizable in o(log k n) time and polynomial work in the model are exactly those in nc k . some improvements to other simulations in the literature of parallel models and reversal complexity are given. *Classify it into one of the five categories: (A) data structures algorithms and theory, computational geometry (B) data structures algorithms and theory, computational complexity (C) data structures algorithms and theory, parallel (D) data structures algorithms and theory, logic (E) artificial intelligence, nlp. The answer can only be (A), (B), (C), (D), or (E).* *### Answer Choice:*
Qwen (7B)	**A**, B, C, D, or E. for example in examples: if example in lines: ans = example
BLOOMz (7B)	(**B**) data structures algorithms and theory, computational complexity</s>
Llama-3 (8B)	**A** ### Explanation: This is a paper abstraction: this paper develops a new and natural parallel vector model, and shows that for all k 1, the languages recognizable in o(log k n) time and polynomial work in the model are exactly those in nc k . some improvements to other simulations in the literature of parallel models and reversal complexity are given.
Baichuan (13B)	The correct choice would be category **A**): Data Structures & Algorithms Theory
Vicuna (13B)	# (**A**) data structures algorithms and theory # (B) data structures algorithms and theory # (C) data structures algorithms and theory # (D) data structures algorithms and theory # (E) data structures algorithms and theory
Encoder-based PLMs	
RoBERTa*+d (123M)	\<Document embedding, embedding of Label A with discrete prompt\> = 0.1953 \<Document embedding, embedding of Label **B** with discrete prompt\> = **0.2074** \<Document embedding, embedding of Label C with discrete prompt\> = 0.2068 \<Document embedding, embedding of Label D with discrete prompt\> = 0.1986 \<Document embedding, embedding of Label E with discrete prompt\> = 0.1919
BERT*+d (110M)	\<Document embedding, embedding of Label A with discrete prompt\> = 0.1936 \<Document embedding, embedding of Label B with discrete prompt\> = 0.2040 \<Document embedding, embedding of Label **C** with discrete prompt\> = **0.2068** \<Document embedding, embedding of Label D with discrete prompt\> = 0.2036 \<Document embedding, embedding of Label E with discrete prompt\> = 0.1919
G2P2+d (63M)	\<Node embedding, embedding of Label A with discrete prompt\> = 0.2021 \<Node embedding, embedding of Label **B** with discrete prompt\> = **0.2245** \<Node embedding, embedding of Label C with discrete prompt\> = 0.2100 \<Node embedding, embedding of Label D with discrete prompt\> = 0.2002 \<Node embedding, embedding of Label E with discrete prompt\> = 0.1633

Table 5: Ablation study with variants of G2P2.

	Cora	Art	Industrial	M.I.
Only $\mathcal{L}_3$	74.66±1.80	52.56±1.09	45.97±0.81	49.05±0.54
Only $\mathcal{L}_2$	77.01±1.30	58.90±0.55	52.99±0.46	59.41±0.85
Only $\mathcal{L}_1$	79.50±1.19	77.37±0.72	78.10±0.34	79.70±0.56
$\mathcal{L}_2+\mathcal{L}_3$	70.04±2.89	49.91±1.57	50.07±0.50	56.14±1.01
$\mathcal{L}_1+\mathcal{L}_3$	79.73±0.89	78.60±0.40	79.97±0.43	80.42±0.45
$\mathcal{L}_1+\mathcal{L}_2$	79.42±1.04	80.55±0.52	81.06±0.33	82.39±0.41
Only label text	79.16±1.23	79.59±0.31	80.86±0.40	81.26±0.36
Random init.	80.03±0.99	80.85±0.43	82.43±0.33	82.64±0.21
G2P2	**80.08**±1.33	**81.03**±0.43	**82.46**±0.35	**82.77**±0.32

Table 6: Analysis of tuning time and parameter size.

	Tuning time per task (in seconds)				Param.
	Cora	Art	Industrial	M.I.	size
RoBERTa	45.47±2.38	64.22±3.62	43.46±2.99	44.99±2.58	123M
RoBERTa*	39.38±2.01	59.56±3.55	35.10±2.75	38.84±2.39	123M
BERT	32.23±1.71	51.77±2.00	31.72±1.77	33.55±2.39	110M
BERT*	34.82±1.68	55.16±2.32	31.11±1.74	29.00±2.23	110M
G2P2	**2.42**±0.41	**22.03**±1.39	**14.63**±1.26	**12.72**±1.17	**2048**

Table 7: Inductive performance on text classification.

	Art	Industrial	M.I.
BERT*	43.66±0.90	48.35±0.25	39.24±0.88
RoBERTa*	69.55±1.14	73.65±0.86	71.96±1.44
G2P2	**79.81**±0.22	**81.29**±0.32	**81.85**±0.33

Hyperparameter study. We first investigate the impact of the interaction coefficient λ in Fig. 6(a), which balances the high-order contrastive losses $(\mathcal{L}_2, \mathcal{L}_3)$. The performance is generally better and stable when λ is slightly bigger (*e.g.*, ≥ 10), indicating the significance of the high-order text-summary and node-summary interactions. Next, we study the prompt length

M in Fig. 6(b), which refers to the number of trainable prompt vectors in Sect. IV-C. The performance is relatively unaffected by the prompt length, and thus it is robust to choose a small

Table 8: Performance on base and unseen classes within the same domain.

Prompts are selected or tuned on the labeled set of the base classes, and tested on the test set of the base classes, or the unseen classes. HM denotes the *harmonic mean* of the testing performance on the base and unseen classes, enabling the assessment of the generalization trade-off between them [87].

(a) Overall

	Base	Unseen	HM
G2P2+d	39.3	38.0	38.4
G2P2	49.7	33.5	39.5
G2P2*	48.7	41.0	43.9

(b) Cora

	Base	Unseen	HM
G2P2+d	28.81±3.35	22.51±1.41	25.14±1.42
G2P2	52.50±1.38	24.83±2.71	33.65±2.72
G2P2*	48.60±1.68	25.70±1.25	33.62±1.46

(c) Art

	Base	Unseen	HM
G2P2+d	40.04±2.35	38.01±0.74	38.95±1.05
G2P2	43.47±2.59	31.97±4.69	36.71±3.73
G2P2*	44.29±2.30	41.43±1.28	42.80±1.61

(d) Industrial

	Base	Unseen	HM
G2P2+d	47.24±2.71	51.80±1.66	49.36±1.64
G2P2	52.84±1.72	39.71±1.66	45.29±0.98
G2P2*	52.44±2.48	50.77±1.74	51.59±2.09

(e) M.I.

	Base	Unseen	HM
G2P2+d	41.12±3.63	39.64±3.88	40.03±0.78
G2P2	49.89±4.76	37.49±3.95	42.43±1.64
G2P2*	49.29±4.25	45.95±1.92	47.47±2.42

Table 9: Continual learning within the same domain.

Prompts are selected or tuned on the labeled data from base classes, and tested on the combined base classes (test set only) and unseen classes.

	Cora	Art	Industrial	M.I.
G2P2+d	18.02±1.70	29.69±1.27	39.27±0.28	27.57±0.61
G2P2	23.53±1.19	27.45±2.33	35.85±1.06	32.17±0.74
G2P2*	28.93±1.42	31.09±0.61	41.00±1.69	34.66±2.03

M (*e.g.*, 4) for efficiency.

Efficiency of prompt tuning. In this experiment, we investigate the prompt tuning efficiency of G2P2 in comparison to the efficiency of traditional fine-tuning. As G2P2 has a transformer component, we compare it with four transformer based models, all of which follow the classical "pre-train, fine-tune" paradigm.

As shown in Tab. 6, prompt tuning in G2P2 is much more efficient than fine-tuning in the baselines, achieving 2.1∼18.8x speedups. The reason is that prompt tuning updates far fewer parameters. In G2P2, we used 4 trainable 512-dimensional prompt vectors, totalling to 2048 parameters only, while fine-tuning in the baselines needs to update the whole pre-trained model with more than 100M parameters. Note that the speedup is not linear w.r.t. the parameter size, due to overheads in the data loader and the optimizer. Overall, our prompt tuning is not only effective under low-resource settings, but also parameter- and computation-efficient.

Inductive ability. Our previous experiments can be deemed transductive as both the pre-training and downstream text classification are conducted on the whole corpus. To further evaluate the generalization ability of G2P2, we adopt an "inductive" setting, whereby we pre-train the text encoder only on a subset of the corpus and perform downstream classification on a disjoint subset. Particularly, in the three Amazon datasets, since user texts have no labels and item texts have labels, it is natural for us to pre-train with only user texts and classify only item texts downstream. We also employ masked language modeling on only the user texts for BERT and RoBERTa, to get BERT* and RoBERTa*. As shown in Tab. 7, G2P2 still performs very well in the inductive setting, illustrating the strong generalization ability of our pre-trained model.

D. Performance and model analyses of G2P2*

Finally, we conduct additional experiments to investigate the performance of G2P2*, in particular the generalization

ability of conditional prompt tuning to handle wider unseen classes. In these experiments, on each dataset we randomly sample some (35 for Cora, 66 for Art, 41 for Industrial, and 35 for M.I.) classes as the base classes, and the same number of classes as unseen classes. Each base class has five labeled instances (5-shot) for selecting or tuning the optimal prompts, and reserves the remaining instances for testing. In contrast, all instances of each unseen class are used for testing, without any labeled data for learning. We compare the three variants of our proposed model: (1) G2P2+d, the zero-shot method with handcrafted discrete prompts chosen based on the base classes; (2) G2P2, which learns continuous and static prompts from the base classes; (3) G2P2*, which learns conditional prompts on the base classes.

In the following, we first conduct experiments to evaluate the within- and cross-domain generalization capability. Then, we investigate the impact of hidden dimension of Meta-net, and the effect of increasing the parameters of G2P2.

Within-domain generalization. Within the same domain, we consider two scenarios of generalization: from base classes to unseen classes, and a continual learning setting where base classes are still included during testing.

Firstly, we report the results on the testing sets of both the base classes and unseen classes in Tab. 8. Note that the base and unseen classes come from the same domain, *i.e.*, the same dataset in our context. Specifically, on the base classes, G2P2+d performs the worst in all cases, showing that the discrete prompts are difficult to select for intrinsically continuous PLMs. In contrast, G2P2 generally obtains the best performance on the base classes, showing that continuous prompt tuning can find better prompts if some labeled data are available for tuning. However, the learned static prompts do not generalize well to the unseen classes, showing a significant class shift between base and unseen classes. Interestingly, G2P2+d performs better on unseen classes than G2P2, showing that the handcrafted prompts are robust to class shifts to some extent. G2P2*, does not achieve the best performance on the base class of the *Cora* dataset but is still the second best, outperforming the third-place G2P2+d by 68.7%. Moreover, we achieve the best results on the new classes. As a result, the harmonic mean of our performance and that of the best-performing G2P2 shows no difference. On the *Industrial* dataset, G2P2* is not the best performer on either the base class or the new class. However,

Table 10: Cross-domain generalization from Amazon to Cora.

No generalization: Prompts are tuned and tested on the labeled data and test set, respectively, of the base classes from the same domain.
Cross-domain generalization: Prompts are tuned in the same way on the source domain, but tested on the unseen classes of the target domain.

	No generalization (base classes)			Cross-domain generalization (unseen classes)		
Source (Tuning)	Art	Industrial	M.I.	Art	Industrial	M.I.
Target (Testing)	Art	Industrial	M.I.	Cora	Cora	Cora
G2P2	43.47±2.59	**52.84**±1.72	**49.89**±4.76	22.83±4.63	20.88±3.99	19.51±5.08
G2P2*	**44.29**±2.30	52.44±2.48	49.29±4.25	**27.68**±3.67	**21.01**±3.94	**27.05**±3.25

it demonstrates balanced performance, ranking just below the best in both categories. Consequently, its overall performance, represented by the harmonic mean, is the highest, surpassing the second-best by 4.5%. Overall, in comparison to G2P2+d and G2P2, G2P2* demonstrates competitive performance on the base classes and excellent performance on the unseen classes, implying that conditional prompt tuning not only fits well to the base classes, but also generalizes to the unseen classes.

Secondly, we address a more pragmatic "continual learning" scenario [88], in which we combine the test sets of both the base and unseen classes during inference. This scenario is designed to evaluate the undesirable effect of catastrophic forgetting of the base classes while adapting to the unseen classes. The results are presented in Tab. 9. It becomes evident that G2P2 loses its competitive edge over G2P2+d, but G2P2* remains superior, implying that G2P2* is robust to continual learning with less "forgetting" of the base classes.

Cross-domain generalization. In the preceding part, we have investigated the ability to generalize to unseen classes within the same domain. In the following, we aim to examine the generalization to unseen classes across different domains. The ability to generalize to out-of-distribution data is a crucial feature in real-world scenarios, and it is intriguing to see whether the handcrafted or learned prompts can withstand shifts across domains. More specifically, we consider cross-domain scenarios within the Amazon ecosystem, and the scenarios with a more challenging shift from Amazon to Cora.

Firstly, we conduct cross-domain experiments within the Amazon ecosystem. Each Amazon dataset (Art, Industrial and M.I.) represents a different domain. As illustrated in Fig. 7, among the three variants, the discrete prompts in G2P2+d exhibit a suboptimal fit on the source domains, yet they demonstrate a reasonable level of generalization across other domains. Conversely, the learnable static prompts in G2P2 fit optimally on the source domains but show minimal generalization ability across other domains. On the other hand, the conditional prompts in G2P2* has performed well on both the source and target domains, showing their excellent performance and generalization capabilities.

Secondly, we consider a more challenging cross-domain scenario, where the source and target domains are from completely different ecosystems. Specifically, we employ the Amazon datasets as the source domains, and aim to generalize to the target domain on Cora. Firstly, the left half of Tab. 10, does not involve cross-domain generalization. In other words, both the source domain and the target domain are from the same domain. Specifically, the source domain is the training set of the dataset, and the target domain is its testing set. The results of
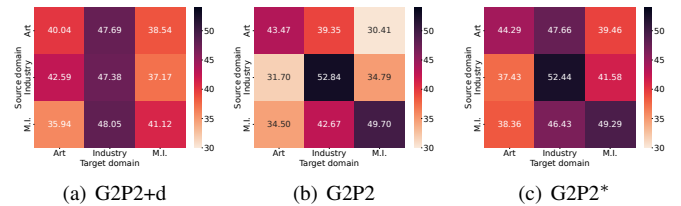

(a) G2P2+d   (b) G2P2   (c) G2P2*

Fig. 7: Cross-domain generalization within Amazon. For each method, prompts are selected or tuned on the labeled data of the base classes from a source domain, and tested on the unseen classes from each target domain.

this part are the few-shot classification results, demonstrating that G2P2*'s few-shot classification capabilities are on par with G2P2. Secondly, the right half of Tab. 10, involves cross-domain generalization. This means that the source and target domains are from different datasets. For instance, the source domain might be the training set of the Art, where we fine-tune G2P2's continuous prompts, or fine-tune both G2P2*'s continuous prompts and the Meta-net component. We then test on the target domain, which is the testing set of another dataset, Cora. This is an extreme test of the model's generalization ability since the training and testing sets are from entirely different datasets. Indeed, such tests also fall under the category of zero-shot classification. From the results in the right half of Tab. 10, we can see that G2P2* performs much better than G2P2 on cross-dataset testing sets, demonstrating the strong generalization capability of G2P2*'s conditional prompts.

Hidden dimension of Meta-net. Recall that our proposed G2P2* incorporates a Meta-net with a hidden layer. We further study the impact of the hidden dimension of the Meta-net, by varying it between 2 and 512. Observations from Fig. 8 indicate that augmenting the hidden dimension, despite expanding the model capacity, does not yield significantly enhanced performance on both base and unseen classes. Generally, a larger model with more parameters performs better, but there's an often overlooked assumption: **sufficient training data**. However, the meta-net of G2P2* was trained under a few-shot setting for the base class, *i.e.*, training samples being extremely scarce. Given this significant lack of training data, simply increasing the model size and parameters can lead to overfitting on base classes. Consequently, the performance on unseen classes declines as the hidden dimension of the meta-net increases. Besides, a larger hidden dimension naturally leads to an increase in computational costs. Consequently, a relatively small value, such as 8, is sufficient yet efficient.

Comparing G2P2* with a bigger G2P2 model. Given that G2P2* incorporates a larger number of parameters than G2P2,
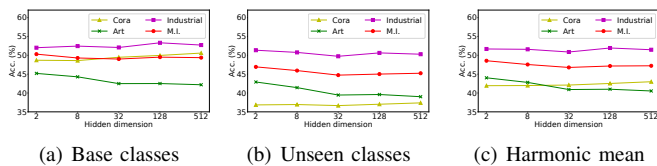
(a) Base classes   (b) Unseen classes   (c) Harmonic mean

Fig. 8: Impact of hidden dimension of Meta-net.

Table 11: G2P2* vs. a bigger G2P2 model on Art.

Model	Param. size Parameters	Base Metric 1	Unseen Metric 2	HM Metric 3
G2P2 ($M = 4$)	2,048	43.47±2.59	31.97±4.69	36.71±3.73
G2P2 ($M = 16$)	**8,192**	40.74±3.31	28.67±2.70	33.64±2.93
G2P2* ($M = 4$)	7,688	**44.29**±2.30	**41.43**±1.28	**42.80**±1.61

specifically through the addition of the Meta-net, it is important to investigate if the observed improvements by G2P2* are merely a consequence of parameter size. Hence, we use more prompt tokens in G2P2 so that its number of parameters becomes comparable to that of G2P2*. Specifically, the number of prompt tokens (M) is increased to 16 from 4 in G2P2, as shown in Tab. 11. The results indicate that merely increasing the parameter size does not lead to performance improvement. On the contrary, compared to $M = 4$, $M = 16$ gives worse performance on both base and unseen classes possibly due to overfitting in low-resource settings. Thus, conditional prompt tuning is a crucial design responsible for the improvements.

VII. CONCLUSION

In this paper, we studied the problem of low-resource text classification. Given that many text documents are related through an underlying network, we proposed a novel model called Graph-Grounded Pre-training and Prompting (G2P2). It consists of three graph interaction-based contrastive strategies in pre-training, and a prompting mechanism for the jointly pre-trained graph-text model in downstream classification. We further extended our model to G2P2* in order to deal with wider unseen classes. Finally, we conducted extensive experiments and showed the advantages of G2P2 and G2P2* in low-resource text classification. We will investigate its application in other text classification scenarios in future work.

ACKNOWLEDGMENTS

REFERENCES

[1] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[2] H. Xu, B. Liu, L. Shu, and P. Yu, "Open-world learning and application to product classification," in *WWW*, 2019, pp. 3413–3419.

[3] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019, pp. 4171–4186.

[4] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," *OpenAI blog*, 2018.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *NeurIPS*, vol. 33, pp. 1877–1901, 2020.

[7] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *ACL*, 2021, pp. 4582–4597.

[8] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *EMNLP*, 2021, pp. 3045–3059.

[9] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *arXiv*, 2021.

[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *TNNLS*, vol. 32, no. 1, pp. 4–24, 2020.

[11] L. Wu, H. Lin, C. Tan, Z. Gao, and S. Z. Li, "Self-supervised learning on graphs: Contrastive, generative, or predictive," *TKDE*, 2021.

[12] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." *ICLR*, vol. 2, no. 3, p. 4, 2019.

[13] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for pre-training graph neural networks," in *ICLR*, 2019.

[14] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "Gpt-gnn: Generative pre-training of graph neural networks," in *KDD*, 2020, pp. 1857–1867.

[15] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *ICML*, 2016, pp. 40–48.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv*, 2013.

[17] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Conditional prompt learning for vision-language models," in *CVPR*, 2022, pp. 16 816–16 825.

[18] Z. Wen and Y. Fang, "Augmenting low-resource text classification with graph-grounded pre-training and prompting," *arXiv*, 2023.

[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*. OpenReview.net, 2017.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[21] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*. OpenReview.net, 2019.

[22] Y. Lu, X. Jiang, Y. Fang, and C. Shi, "Learning to pre-train graph neural networks," in *AAAI*, vol. 35, no. 5, 2021, pp. 4276–4284.

[23] X. Cao, Y. Shi, H. Yu, J. Wang, X. Wang, Z. Yan, and Z. Chen, "Dekr: description enhanced knowledge graph for machine learning method recommendation," in *SIGIR*, 2021, pp. 203–212.

[24] H. Linmei, T. Yang, C. Shi, H. Ji, and X. Li, "Heterogeneous graph attention networks for semi-supervised short text classification," in *EMNLP*, 2019, pp. 4821–4830.

[25] Y. Liu, S. Yang, C. Lei, G. Wang, H. Tang, J. Zhang, A. Sun, and C. Miao, "Pre-training graph transformer with multimodal side information for recommendation," in *ACMMM*, 2021b, pp. 2853–2861.

[26] J. Zhao, M. Qu, C. Li, H. Yan, Q. Liu, R. Li, X. Xie, and J. Tang, "Learning on large-scale text-attributed graphs via variational inference," in *ICLR*, 2023.

[27] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang *et al.*, "Pre-trained models: Past, present and future," *AI Open*, vol. 2, pp. 225–250, 2021.

[28] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *NeurIPS*, vol. 32, 2019.

[29] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized bert pretraining approach," *arXiv*, 2019.

[30] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer." *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[31] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *arXiv*, 2021.

[32] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "Auto-Prompt: Eliciting knowledge from language models with automatically generated prompts," in *EMNLP*, 2020, pp. 4222–4235.

[33] T. Schick and H. Schütze, "Exploiting cloze-questions for few-shot text classification and natural language inference," in *EACL*, 2021, pp. 255–269.

[34] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," in *ACL*, 2021, pp. 3816–3830.

[35] G. Qin and J. Eisner, "Learning how to ask: Querying lms with mixtures of soft prompts," in *NAACL*, 2021, pp. 5203–5212.

[36] Z. Zhong, D. Friedman, and D. Chen, "Factual probing is [MASK]: Learning vs. learning to recall," in *NAACL*, 2021, pp. 5017–5033.

[37] S. Hu, N. Ding, H. Wang, Z. Liu, J. Wang, J. Li, W. Wu, and M. Sun, "Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification," in *ACL*, 2022, pp. 2225–2240.

[38] S. Min, M. Lewis, H. Hajishirzi, and L. Zettlemoyer, "Noisy channel language model prompting for few-shot text classification," in *ACL*, 2022, pp. 5316–5330.

[39] Y. Sun, Y. Zheng, C. Hao, and H. Qiu, "NSP-BERT: A prompt-based zero-shot learner through an original pre-training task–next sentence prediction," *arXiv*, 2021.

[40] W. Zhang, Y. Deng, X. Li, Y. Yuan, L. Bing, and W. Lam, "Aspect sentiment quad prediction as paraphrase generation," in *EMNLP*, 2021, pp. 9209–9219.

[41] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, "PTR: Prompt tuning with rules for text classification," *arXiv*, 2021.

[42] Z. Tan, X. Zhang, S. Wang, and Y. Liu, "MSP: Multi-stage prompting for making pre-trained language models better translators," in *ACL*, 2022, pp. 6131–6142.

[43] X. Chen, N. Zhang, X. Xie, S. Deng, Y. Yao, C. Tan, F. Huang, L. Si, and H. Chen, "KnowPrompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction," in *WWW*, 2022, pp. 2778–2788.

[44] O. Sainz, O. L. de Lacalle, G. Labaka, A. Barrena, and E. Agirre, "Label verbalization and entailment for effective zero and few-shot relation extraction," in *EMNLP*, 2021, pp. 1199–1212.

[45] M. Sun, K. Zhou, X. He, Y. Wang, and X. Wang, "GPPT: Graph pre-training and prompt tuning to generalize graph neural networks," in *KDD*, 2022, pp. 1717–1727.

[46] Z. Tan, R. Guo, K. Ding, and H. Liu, "Virtual node tuning for few-shot node classification," *arXiv*, 2023.

[47] T. Fang, Y. Zhang, Y. Yang, and C. Wang, "Prompt tuning for graph neural networks," *arXiv*, 2022.

[48] Z. Liu, X. Yu, Y. Fang, and X. Zhang, "GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks," in *WWW*, 2023, pp. 417–428.

[49] T. Miyato, A. M. Dai, and I. J. Goodfellow, "Adversarial training methods for semi-supervised text classification," in *ICLR*. OpenReview.net, 2017.

[50] Q. Xie, Z. Dai, E. Hovy, T. Luong, and Q. Le, "Unsupervised data augmentation for consistency training," *NeurIPS*, vol. 33, pp. 6256–6268, 2020.

[51] J. Chen, Z. Yang, and D. Yang, "MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification," in *ACL*, 2020, pp. 2147–2157.

[52] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, pp. 1126–1135.

[53] M. Yu, X. Guo, J. Yi, S. Chang, S. P. Y. C. G. Tesauro, H. W. B. Zhou, and A. Foundations-Learning, "Diverse few-shot text classification with multiple metrics," in *NAACL*, 2018, pp. 1206–1215.

[54] X. Han, H. Zhu, P. Yu, Z. Wang, Y. Yao, Z. Liu, and M. Sun, "FewRel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation," in *EMNLP*, 2018, pp. 4803–4809.

[55] T. Bansal, R. Jha, T. Munkhdalai, and A. McCallum, "Self-supervised meta-learning for few-shot natural language classification tasks," in *EMNLP*, 2020, pp. 522–534.

[56] Y. Bao, M. Wu, S. Chang, and R. Barzilay, "Few-shot text classification with distributional signatures," in *ICLR*, 2020.

[57] F. Zhou, C. Cao, K. Zhang, G. Trajcevski, T. Zhong, and J. Geng, "Meta-GNN: On few-shot node classification in graph meta-learning," in *CIKM*, 2019, pp. 2357–2360.

[58] N. Wang, M. Luo, K. Ding, L. Zhang, J. Li, and Q. Zheng, "Graph few-shot learning with attribute matching," in *CIKM*, 2020, pp. 1545–1554.

[59] Z. Wen, Y. Fang, and Z. Liu, "Meta-inductive node classification across graphs," in *SIGIR*, 2021, pp. 1219–1228.

[60] Z. Wen, "Generalizing graph neural network across graphs and time," in *WSDM*, 2023, pp. 1214–1215.

[61] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021, pp. 8748–8763.

[62] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *IJCV*, vol. 130, no. 9, pp. 2337–2348, 2022.

[63] Z. Wang, J. Wang, Y. Guo, and Z. Gong, "Zero-shot node classification with decomposed graph prototype network," in *KDD*, 2021, pp. 1769–1779.

[64] J. Liu, Z. He, L. Wei, and Y. Huang, "Content to node: Self-translation network embedding," in *KDD*, 2018, pp. 1794–1802.

[65] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," *NeurIPS*, vol. 29, 2016.

[66] Y. Zhang, H. Jiang, Y. Miura, C. D. Manning, and C. P. Langlotz, "Contrastive learning of medical visual representations from paired images and text," *arXiv*, 2020.

[67] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin, "Joint embedding of words and labels for text classification," in *ACL*, 2018, pp. 2321–2331.

[68] X. Liu, K. Ji, Y. Fu, W. Tam, Z. Du, Z. Yang, and J. Tang, "P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks," in *ACL*, 2022, pp. 61–68.

[69] D. Ha, A. M. Dai, and Q. V. Le, "Hypernetworks," in *ICLR*, 2017.

[70] Y. Wu and T. Lee, "Reducing model complexity for dnn based large-scale audio classification," in *ICASSP*, 2018, pp. 331–335.

[71] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," *arXiv*, 2021.

[72] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *EMNLP*, 2019, pp. 188–197.

[73] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017.

[74] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *AAAI*, vol. 33, no. 01, 2019, pp. 7370–7377.

[75] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: a good embedding is all you need?" in *ECCV*, 2020, pp. 266–282.

[76] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang *et al.*, "Qwen technical report," *arXiv*, 2023.

[77] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. L. Scao, M. S. Bari, S. Shen, Z.-X. Yong, H. Schoelkopf *et al.*, "Crosslingual generalization through multitask finetuning," *arXiv*, 2022.

[78] "Llama 3 model card," *https://llama.meta.com/docs/model-cards-and-prompt-formats/meta-llama-3/Llama 3 Model Card*, 2024.

[79] "A 13b large language model developed by baichuan intelligent technology." *https://github.com/baichuan-inc/Baichuan-13B*, 2024.

[80] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing *et al.*, "Judging LLM-as-a-judge with MT-bench and chatbot arena," *NeuraIPS*, vol. 36, 2024.

[81] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "GPT-4 technical report," *arXiv*, 2023.

[82] "An overview of bard: an early experiment with generative ai," *https://ai.google/static/documents/google-about-bard.pdf*, 2024.

[83] "Alpaca: A strong, replicable instruction-following model," *https://crfm.stanford.edu/2023/03/13/alpaca.html*, 2024.

[84] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *ACL*, 2016, pp. 1715–1725.

[85] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *KDD*, 2014, pp. 701–710.

[86] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *EMNLP*, 2020, pp. 38–45.

[87] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning-the good, the bad and the ugly," in *CVPR*, 2017, pp. 4582–4591.

[88] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural networks*, vol. 113, pp. 54–71, 2019.

Zhihao Wen received his Ph.D. degree in Computer Science from the School of Computing and Information Systems, Singapore Management University. His research interests include graph-based machine learning and data mining, as well as their applications for the Web and social media.

Yuan Fang received his Ph.D. degree in Computer Science from University of Illinois at Urbana Champaign in 2014. He is currently an Assistant Professor in the School of Computing and Information Systems, Singapore Management University. His research focuses on graph-based machine learning and data mining, as well as their applications for the Web and social media.