Prompt Learning for Textual Heterogeneous Information Networks

YANG FANG, National University of Defense Technology, Changsha, China
XIANG ZHAO, National Key Laboratory for Big Data and Decision, National University of Defense Technology, Changsha, China
DAOJIAN ZENG, Hunan Normal University, Changsha, China
WEIDONG XIAO, Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China
MAARTEN DE RIJKE, University of Amsterdam, Amsterdam, The Netherlands
MINNAN LUO, Xi'an Jiaotong University, Xi'an, China
XINWANG LIU, National University of Defense Technology, Changsha, China

Heterogeneous information networks (HINs) play an indispensable role in a wide range of domain-specific applications, from recommender systems to conversational platforms. Textual HINs are graphs with abundant textual information. Currently, most advanced approaches to mine textual features from HINs follow a "pre-training and fine-tuning" schema, which may cause a "negative transfer" problem since there is a gap between pre-training tasks and downstream tasks. We propose a prompt-learning framework P-HIN that provides a new angle to align textual information and graph information, while narrowing down the gap between the pre-trained models and various downstream tasks. To the best of our knowledge, we are among the first to introduce and exploit the idea of prompt learning to align HIN features and textual features. The proposed framework P-HIN is composed of a text encoder and a graph encoder and uses contrastive learning to align and fuse the graph–text pair. This pre-training operation naturally fits the few-shot learning setting. For the graph encoder, we introduce two graph pre-training tasks, masked node modeling and edge reconstruction, to exploit self-supervised information. During optimization, instead of handcrafted prompts, we use a learnable continuous text that enables more efficient and task-relevant transfer to downstream datasets. We consider a residual connection to use the context from the graph to prompt the text encoder. In experiments, P-HIN consistently and significantly outperforms state-of-the-art alternatives on all real-life datasets.

This work was partially supported by National Natural Science Foundation of China (Nos. 62306322, U25B2047, 62272469), and The Science and Technology Innovation Program of Hunan Province (No. 2023RC1007), the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.1389.20.183, and KICH3.LTP.20.006, and the European Union under grant agreement No. 101201510 (UNITE). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.
Authors' Contact Information: Yang Fang (corresponding author), National University of Defense Technology, Changsha, China; e-mail: fangyang12@nudt.edu.cn; Xiang Zhao, National Key Laboratory for Big Data and Decision, National University of Defense Technology, Changsha, China; e-mail: xiangzhao@nudt.edu.cn; Daojian Zeng, Hunan Normal University, Changsha, China; e-mail: zengdj@hunnu.edu.cn; Weidong Xiao, Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China; e-mail: wdxiao@nudt.edu.cn; Maarten de Rijke, University of Amsterdam, Amsterdam, The Netherlands; e-mail: m.derijke@uva.nl; Minnan Luo, Xi'an Jiaotong University, Xi'an, China; e-mail: minnluo@xjtu.edu.cn; Xinwang Liu, National University of Defense Technology, Changsha, China; e-mail: xinwangliu@nudt.edu.cn.
*For correspondence, please contact Xiang Zhao.

ACM 1558-2868/2026/9-ART158
https://doi.org/10.1145/3831697

CCS Concepts: • **Information systems** → **Network data models**;

Additional Key Words and Phrases: Textual Heterogeneous Information Network, Prompt Learning

ACM Reference format:
Yang Fang, Xiang Zhao, Daojian Zeng, Weidong Xiao, Maarten de Rijke, Minnan Luo, and Xinwang Liu. 2026. Prompt Learning for Textual Heterogeneous Information Networks. *ACM Trans. Inf. Syst.* 44, 7, Article 158 (September 2026), 25 pages.
https://doi.org/10.1145/3831697

1 Introduction

Heterogeneous Information Networks (HINs) are widespread and can be found in diverse contexts [38]. Real-world networks, which span **Knowledge Graphs (KGs)**, social networks, and user–item interactions in search and recommender systems, are inherently heterogeneous, containing a variety of node and link types [14, 26, 32]. Unlike homogeneous information networks, where all nodes are assumed to be of the same type, HINs offer a more diverse and nuanced way to represent and describe network structures. Textual HINs are graphs with textual information, e.g., a title and an abstract of a paper node in an academic network, which can provide fruitful additional information for downstream tasks. This enables the development of more effective solutions for a broad range of tasks in knowledge discovery, information retrieval, and data mining, such as link prediction, node classification, recommendation systems, and anomaly detection [2, 3, 5, 49].

Most current work on HINs ignores such textual information and maps the node of a graph into low-dimensional representations based only on structural information [10, 21, 24]. To address this gap, some models for mining information networks propose to integrate textual information into the node representations. They mainly design a framework to incorporate a node's structural information with textual information to generate a single representation [39, 43, 44].

The textual network embedding models discussed above rely heavily on the availability of a large number of training labels to function effectively. When training, it is typically assumed that most of the labels in the network are accessible, and these models are designed to classify nodes only based on the labels they have been trained on.

To enable models to deal with few-shot settings, the "pre-training and fine-tuning" paradigm has been proposed [20, 48]. Here, models are first pre-trained in self-supervised manner, which does not need labels, followed by transfer learning of the pre-trained model to classify nodes with unseen labels after the fine-tuning stage. Such "pre-training and fine-tuning" methods may have "negative transfer" issues. This is because downstream tasks involving graphs may be very diverse, including node level (e.g., node classification), graph level (e.g., link prediction), and edge level (e.g., link prediction) tasks. Thus, gaps may emerge when pre-trained models are fine-tuned on the downstream tasks.

To address the above issues, prompt learning in **Natural Language Processing (NLP)** has been proposed. This strategy reformulates downstream tasks to look like pre-training tasks [27]. With or without slight fine-tuning, prompt learning helps to quickly adapt prior knowledge to unseen tasks, which narrows the gap between pre-training and downstream tasks, while facilitating few-shot learning. Recently, prompt learning has also been adopted in multi-modal scenarios, aligning image and textual data [30, 52]. However, no technique based on prompt learning has been used to deal with graph and textual data.

Some work apply prompt learning to graph data [7, 28, 34, 35]. However, they mainly treat the graph as natural language, adding prompt tokens or prompt subgraphs. They focus on mining the features of the network itself and are not able to use potentially useful textual information.

Both the existing "pre-training and fine-tuning" models for textual networks and "pre-training, prompt learning, and fine-tuning" models for normal networks have originally been designed for single-modal information networks. No prior research has attempted to address the dual challenges of multi-modal and few-shot learning within the context of textual HINs.

Our Approach. We address the limitations highlighted above and propose

- —a text encoder to use textual information,
- —a graph encoder that encodes structural and heterogeneous features, along with self-supervised information,
- —a contrastive learning mechanism to fuse text and graph embeddings, and
- —a learnable continuous prompt learning framework to align textual and graph data, while solving the few-shot problem on textual HINs.

We refer to the aforementioned prompt learning framework, which we have developed to tackle few-shot learning problems on textual HINs, as P-HIN. To the best of our knowledge, our work is among the *first* to apply prompt learning to integrate graph data with textual data and benefit downstream tasks.

P-HIN consists of two key components, a text encoder and a graph encoder, which are used for transforming text and graph data into low-dimensional vector representations, respectively. We adopt **Sentence-BERT (SBERT)** [31] as the text encoder to produce the textual embeddings. For our graph encoder, we first sample subgraphs to be processed and force them to have all types of nodes to ensure heterogeneity. Then, we apply an autoencoder [40] mechanism to explore structural features and a Bi-LSTM on type-grouped nodes to capture the heterogeneity of a graph. Additionally, we introduce two graph pre-training tasks, viz. **Masked Node Modeling (MNM)** and **Edge Reconstruction (ER)**, to effectively use node-level and edge-level self-supervised information.

The natural question that emerges at this stage is: *how to integrate the learned text embedding and graph embedding into one single representation*? We introduce a contrastive learning framework that is able to fuse these two kinds of embedding. Given a pair of text and subgraph, if they both belong to a given node, they are matched. A contrastive learning framework is designed to maximize the similarity score of a matched pair while minimizing the score of an unmatched pair.

We need to transfer the above pre-trained model to downstream tasks to fit the few-shot setting. During the optimization phase, for each new classification task, the classification weights can be generated by feeding prompts describing classes of interest to the text encoder and comparing them with the structural and heterogeneous features generated by the graph encoder.

Prompt design is crucial to performance on downstream tasks, as a slight change of words in the prompts may influence the model performance [52]. Instead of designing handcrafted prompts like "a paper of [CLASS] domain," we introduce learnable and continuous prompts that are generated automatically. As we will see below, our automated prompt engineering leads to more task-relevant and efficient transfer for our pre-trained model.

Experiments. To demonstrate the effectiveness of P-HIN, we consider four real-life datasets. OAG is a bibliographic network with papers to be classified into research domains. YELP is a venue check-in dataset with restaurants to be categorized into different kinds of labels. These two datasets only have five labels. We also consider Amazon and Reddit, an e-commerce network with 16 labels and a forum dataset with 41 labels, to verify that P-HIN can be applied in diverse practical scenarios. Our work is applicable to four downstream tasks: node classification, graph classification, link prediction, and keyword generation, showcasing its adaptability and wide-ranging applicability. Keyword generation is the task that cannot be realized by traditional textual network embedding methods. P-HIN is able to generate keywords thanks to the autoregressive generation abilities of

the prompt-tuning framework. We observe that P-HIN consistently and significantly surpasses state-of-the-arts across all datasets and downstream tasks we evaluated.

Contribution. Our contribution is three-fold:

- —We propose a prompt learning framework P-HIN to align textual information and graph information in textual HINs, and deal with few-shot learning problems simultaneously.
- —We introduce a graph encoder that can capture structural and heterogeneous features of HINs, meanwhile preserving the node-level and edge-level self-supervised information.
- —We demonstrate that P-HIN significantly and consistently outperforms state-of-the-art alternatives across various textual HINs.

Organization. Section 2 introduces related work, and Section 3 introduces some key preliminaries. Section 4 provides a detailed account of the P-HIN framework. Section 5 presents our experimental setup, and Section 6 details and analyses our experimental results. Section 7 concludes the article.

2 Related Work

Representation Learning for HINs. Traditional representation learning models for HIN are based on the meta structure, which depicts the heterogeneous features, i.e., metapath or metagraph. metapath2vec [6] uses a heterogeneous SkipGram to describe the metapath. HINE [16] employs a metapath-based proximity measure that minimizes the discrepancy between the joint probability of nodes and their empirical probabilities. HIN2Vec [11] uses Hadamard multiplication of nodes and metapaths to encode and capture heterogeneous features within the network. M-HIN [9] and M-DHIN [10] use metagraphs in complex space to encode multiple heterogeneous relations.

Graph neural networks have been used to handle heterogeneous features. HetGNN [50] encodes type-specific features via Bi-LSTM and then fuses the features of different types. HGT [15] describes heterogeneous attention using node- and edge-type-dependent parameters. SeHGNN [45] employs a lightweight aggregator to pre-compute neighbor information and a transformer-based semantic fusion module to integrate features from various metapaths. ie-HGCN [46] introduces a hierarchical aggregation framework, which first performs object-level aggregation and then proceeds with type-level aggregation. HeCo [42] uses a cross-view contrastive mechanism, using both network schema and meta-path views, to capture both local and high-order structures within a HIN.

Several methods have been developed specifically for pre-training HINs. CPT-HG [18] introduces relation- and subgraph-level pre-training tasks and employs contrastive learning to improve the quality of learned embeddings. PT-HGNN [17] designs node- and schema-level pre-training tasks to contrastively preserve both heterogeneous semantic and structural features. PF-HIN [8] transforms the neighborhood into sequences and creates node- and sequence-level pre-training tasks. SHGP [47] consists of two modules sharing a single attention-aggregation mechanism; one module generates pseudo-labels through structural clustering, which serve as self-supervision signals to guide the other module in learning network embeddings.

The above training or pre-training methods designed for representing HINs all focus on the graph itself while neglecting the textual information.

Training for Textual Network. Many graphs have rich textual information [44], which raises the question of computing textual network embeddings [39]. TADW [44] incorporates textual features into the representation using a matrix factorization framework. CENE [36] treats texts as nodes to integrate textual and structural information. CANE [39] learns a text-aware node embedding through a mutual attention mechanism that models the semantics of nodes. WANE [33] enhances node representations by integrating textual features, achieved through matching important words between text sequences for all node pairs. NEIFA [43] introduces a deep neural architecture designed

to effectively combine structural and textual information into a unified representation. DetGP [4] proposes a Gaussian Process to dynamically model the structural and textual information.

Some researchers have also incorporated the schema of pre-training and fine-tuning into textual network processing. Unlike traditional textual network embedding models that focus on learning node representations for downstream tasks, these approaches primarily pre-train language models to be fine-tuned for specific applications. For example, DRAGON [48] takes pairs of text segments and relevant KG subgraphs as input and bidirectionally fuses information from both modalities. It employs two pre-training tasks, viz. masked language modeling and link prediction. PATTON [20] introduces two pre-training strategies, namely network-contextualized masked language modeling and masked node prediction, aiming to capture the inherent dependencies between textual attributes and network structures.

The aforementioned models that adopt the "pre-training and fine-tuning" schema may encounter a "negative transfer" issue, as graph tasks, including node-, graph-, and edge-level tasks, are highly diverse, and the pre-training tasks often do not align well with downstream tasks. P-HIN aims to offer a fresh perspective for exploring textual networks and to bridge the gap between pre-training and downstream tasks by incorporating prompting techniques.

Prompt Learning. Cloze-style prompts have become a widely used technique in NLP research. This approach uses pre-trained language models to generate answers, thereby enhancing performance across various downstream NLP tasks. The design of these prompts is crucial for achieving optimal results. For instance, Jiang et al. [19] employ text mining and paraphrasing to create candidate prompts and select the best ones based on training accuracy. However, crafting discrete prompts often requires expert input, and there is no guarantee that these prompts will be optimal. Therefore, another line of work [22, 25, 51] adopts learnable continuous prompts, all of which optimize prompt vectors in the word embedding space. The technique is widely used in NLP. See [27] for a comprehensive survey.

Integrating text and graph data could be regarded as a multi-modal problem. Multi-modal prompt learning models [30, 52] focus on aligning the image and text in a contrastive learning framework.

More and more studies of prompt learning on graphs have been put forward recently. Wang et al. [41] develop a theoretical framework that rigorously examines graph prompting within the context of data operations, explaining why graph prompt learning works. Zi et al. [53] provide a comprehensive benchmark for graph prompt learning, integrating six pre-training strategy and five prompt techniques. GPPT [34] applies prompt learning on graph data by adjusting the downstream node classification task to look similar as its pre-training task. GraphPrompt [28] unifies pre-training and downstream tasks into a common task template and introduces prompt tuning to enhance downstream task performance. However, it focuses solely on graph data, neglecting textual information and failing to address the heterogeneity of graphs. Sun et al. [35] propose a novel multi-task prompting method for graph models, unifying the format of graph prompts and language prompts through prompt tokens, token structures, and insertion patterns. GPF [7] operates on the input graph's feature space and theoretically achieves an equivalent effect to any form of prompting function. HetGPT [29] applies the "pretrain-prompt" framework on HINs, while it is limited to one downstream task, i.e., node classification. EdgePrompt [12] chooses to learn edge-specific prompt vectors and incorporates edge prompts through message passing in pre-trained GNNs, while it is limited to node classification and graph classification tasks. IA-GPL [23] designs instance-specific prompts and employs learnable codebook vectors to quantize these prompts, while it is also limited to node classification and graph classification tasks. See [37] for a comprehensive survey.

The aforementioned graph prompting methods primarily focus on applying prompting techniques to the structural aspects of HINs, overlooking the rich textual information available. P-HIN is the first work to introduce prompting techniques for aligning and fusing graph and textual data within

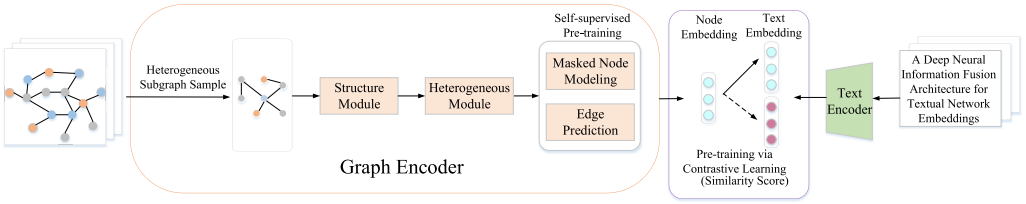


Fig. 1. The pre-training framework of P-HIN.

textual HINs, enabling multi-modal few-shot learning. This approach also enhances performance across various downstream tasks, including node and graph classification, link prediction, and keyword generation.

3 Preliminaries

Textual HIN. A textual HIN is denoted as $G = (V, E, R, T)$, where V denotes the node set, E denotes the edge set, and R denotes the text set. Each $v_i \in V$ is associated with some textual information $r_{v_i} \in R$. T_V denotes the set of node type and T_E denotes the set of edge type. A HIN is defined as a network where $|T_V| > 1$ and/or $|T_E| > 1$, otherwise, it is a homogeneous network.

Prompt Learning for Textual HINs. Given a textual HIN $G = (V, E, R, T)$, the task involves pre-training both the HIN and textual information, and then aligning them via prompting techniques to form the adapted language model M_G. Then, we fine-tune the language model M_G on different downstream tasks in a few-shot setting, not only for typical graph tasks such as node classification, graph classification, and link prediction, but also for NLP tasks, such as keyword generation.

4 The Proposed Model P-HIN

4.1 Text Encoder

The pre-training framework of P-HIN is illustrated in Figure 1. It is composed of two encoders, i.e., a text encoder and a graph encoder. The text encoder maps natural language text to a low-dimensional representation. We use the SBERT [31] model to generate the fixed-size text embeddings.

Instead of using a pre-trained text model, we train the text encoder from scratch using the textual information in HIN. Using a pre-trained text model would introduce external corpora, which would be unfair for existing models processing textual HINs.

4.2 Graph Encoder

The graph encoder maps graph data into a low-dimensional embedding.

Heterogeneous Subgraph Sample. For a given node, we first need to sample the subgraph around it. Notice that only one subgraph will be sampled for a given node. Then, the subgraph is processed by the graph encoder to generate the node representation. After sampling the subgraph, the nodes within a subgraph will be ranked via centrality metrics that evaluate a node's importance [1]. The rank of each node will be used as position information in the following pipeline.

The sampling strategy we employ is **Random Walks with Restart (RWR)**. This strategy starts a random walk from the given node v. However, RWR has a distinctive feature: at each step, before moving, the walker has a fixed probability p of being teleported back to the original starting node. In other words, it will either hop to a neighbor or restart at the source. This process continues until all types of the nodes are collected, so as to ensure the encoder captures the heterogeneity of the network. The restart probability forces the walk to stay relatively close to the source, capturing the local neighborhood structure. This makes RWR particularly effective at finding nodes that are not

only directly connected but also contextually similar to the source based on the overall network topology.

Structural Module. We adopt an autoencoder to capture the structural information of the subgraph. Given the adjacency matrix A of a subgraph, it will first be processed by an encoder to generate the latent embeddings by several layers. The decoder then reverses this process to generate the reconstructed output $\hat{A}$. The training process for an autoencoder aims at minimizing the discrepancy between its input and output data, ensuring that nodes with similar structural properties are mapped to similar embeddings. Mathematically,

$$\mathcal{L}_{\text{structure}} = \|(\hat{A} - A) \odot B\|_F^2, \tag{1}$$

in which B is the penalty assigned to non-zero elements to alleviate the sparsity issue; $\odot$ is element-wise product.

Heterogeneous Module. Nodes sharing the same type are first grouped together so as to capture the heterogeneity of a graph. Although this grouping may disrupt the original subgraph structure, the structural features are already preserved by the autoencoder in the structural module. To process the distinct features within each group, we employ a Bi-LSTM, which is effective at capturing the interactions among node features, thereby providing strong expressive power for complex pattern representation.

Given a node group $S_{T_j}^v$ with type T_j, the representation $h_{T_j}^v$ of node v can be calculated as:

$$h_{T_j}^v = \frac{\sum^{S_{T_j}^v} \text{Bi-LSTM}\{v\}}{|S_{T_j}^v|}. \tag{2}$$

Next, we apply an attention mechanism to aggregate information from all type groups, generating a comprehensive representation h_v for the given node.

To generate the representation of the given node, we use an attention mechanism to aggregate the information from all type groups:

$$h_v = \sum_{T_j \in \{T\}} \alpha^{v,j} h_{T_j}^v, \tag{3}$$

$$\alpha^{v,j} = \frac{\exp\{\delta(u^{\mathrm{T}} h_{T_j}^v)\}}{\sum_{T_i \in \{T\}} \exp\{\delta(u^{\mathrm{T}} h_{T_i}^v)\}}, \tag{4}$$

in which δ represents the activation function, specifically a LeakyReLU, and $u \in \mathbb{R}^d$ represents the attention parameter.

Self-Supervised Pre-Training. To further pre-train the subgraphs using self-supervised information, we introduce two pre-training tasks: MNM and ER, which enable node-level and edge-level exploration of the graph, respectively.

For MNM, we sort the nodes in the subgraph based on their rank and we randomly replace 15% of the nodes with [MASK] token. The sorted sequential nodes will be fed into a transformer encoder, in which the embeddings produced by a Bi-LSTM will be taken as the token embeddings and the rank information is taken as the position embedding. The learned hidden state h_v^{Trans} after the transformer encoder will be fed into a feedforward layer to predict the target node. Mathematically,

$$z_v = \text{Feedforward}(h_v^{Trans}), \tag{5}$$

$$\mathbf{p_v} = \text{softmax}(\mathbf{W}^{\text{MNM}} z_v), \tag{6}$$

in which z_v represents the output of the feedforward layer, $\mathbf{W}^{\text{MNM}} \in V_v \times d$ is the classification weight shared with the input node embedding matrix, d represents the dimension of the hidden state

size, V_v represents the number of nodes in the subgraph, and $\mathbf{p_v}$ denotes the predicted probability distribution of node v over all nodes. The loss is computed using the cross-entropy between the predicted distribution $\mathbf{p}_i^t$ and the one-hot label $\mathbf{y}_i^t$, ensuring the model accurately predicts the masked nodes,

$$\mathcal{L}_{\text{MNM}} = -\sum_i y_i \log p_i, \tag{7}$$

in which y_i and p_i represent the ith components of the one-hot label vector $\mathbf{y}_i$ and the predicted probability distribution vector $\mathbf{p}_i$, respectively.

For ER, we sample the positive and negative edges in a subgraph. Positive edges do exist in the original subgraph while the negative edges do not exist in the original subgraph. Given the positive edges and negative edges united as E_S, in practice, we set $|E_S| = 6$ with positive edges and negative edges split equally. We calculate the score of ER via the inner product between the pair of nodes, i.e., $\hat{e}_{uv} = h_v \odot h_u$. To compute the ER loss, we adopt the binary cross-entropy loss between the predicted edges and ground-truth edges:

$$\mathcal{L}_{\text{ER}} = -\frac{1}{|E_S|} \sum_{e_{uv} \in |E_S|} \text{BinaryCrossEntropy}(e_{uv}, \hat{e}_{uv}). \tag{8}$$

4.3 Pre-Training via Contrastive Learning

P-HIN is trained to fuse the representation spaces of text and graphs, while the learning objective is designed as a contrastive loss. Following standard practices in contrastive learning, given a batch of text-graph pairs, P-HIN needs to maximize the similarity scores for matched pairs, while minimizing the scores for unmatched pairs. We adopt cosine similarity, mathematically,

$$\mathcal{L}_{\text{contrast}} = -\log \frac{e^{\chi(h_q, h_i^+)}}{\sum_{i=1}^{N} e^{\chi(h_q, h_i)}}, \tag{9}$$

where $\chi()$ denotes the similarity scores, h_q denotes the targeted text representation, h^i denotes the node representation, while h_i^+ denotes the matched one.

The overall pre-training loss is combined with the structural loss, MNM loss, ER loss, and contrastive loss sharing equal weights:

$$\mathcal{L}_{\text{Overall-loss}} = \mathcal{L}_{\text{structure}} + \mathcal{L}_{\text{MNM}} + \mathcal{L}_{\text{ER}} + +\mathcal{L}_{\text{contrast}}. \tag{10}$$

In a contrastive learning setting, high-quality negative samples are helpful to boost the model performance. The text and subgraphs we use in a batch are chosen from nodes with the same label to make them hard to be distinguished.

4.4 Prompt-Tuning Optimization

Figure 2 illustrates the prompt-tuning optimization framework. P-HIN can be applied in a few-shot setting as it is pre-trained to predict whether a node's subgraph matches a textual description. This can be realized by comparing the node embedding generated by the graph encoder with the classification weights produced by the text encoder. The textual description can be used to specify node classes of interest. Given a node v, its node embedding learned by the graph encoder is denoted as H, and the weight vectors generated by the text encoder is denoted as $\{w_i\}_{i=1}^K$, where K denotes the number of classes. Each weight w_i is learned from the prompt, e.g., "a paper of [CLASS] domain," and the class token can be the specific class name, such as "information retrieval," "database," or "data mining." For a graph classification downstream task, the prompt is designed similar as for node classification, that is, "a graph of [CLASS] community." To facilitate the link prediction downstream task, the prompt could also be designed as "The two nodes are [CLASS],"

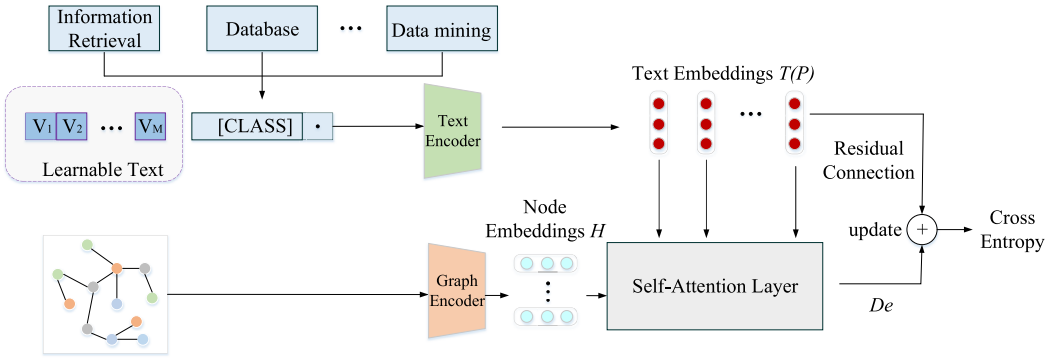


Fig. 2. The prompt-tuning optimization framework of P-HIN.

and it is a binary class token like "connected" and "unconnected." Mathematically, the prediction probability can be calculated as:

$$p(y = i|v) = \frac{\exp(\langle w_i, H\rangle/\tau)}{\sum_{j=1}^{K} \exp(\langle w_j, H\rangle/\tau)}, \tag{11}$$

in which τ is a learnable temperature parameter. It is optimized jointly with other parameters during training to adaptively scale the similarity logits and balance the concentration of the contrastive loss. $\langle \cdot, \cdot \rangle$ denotes the similarity score.

Continuous Prompt. Instead of manual prompts that should be designed by experts, here we choose continuous vectors to replace the text words, which could be learned end-to-end from data. Specifically, the prompt P sent to the text encoder should be devised as:

$$P = [V_1][V_2] \dots [V_M][\text{CLASS}], \tag{12}$$

where $[V_M]$ is a vector with the same dimension as the word embeddings in the training phase, and M is a hyper-parameter that denotes the number of continuous vectors in the prompt. After sending the continuous prompt P to the text encoder $Text(\cdot)$, the classification weight vector representing a node's concept can be obtained. Mathematically, the prediction probability is calculated as:

$$p(y = i|v) = \frac{\exp(\langle Text(P_i), H\rangle)/\tau}{\sum_{j=1}^{K} \exp(\langle Text(P_j), H\rangle/\tau)}, \tag{13}$$

where the class token within each prompt P_i is replaced by the word embeddings of the ith class name.

Residual Connection. Considering the context nodes of the given node, e.g., the author nodes of the paper node, will help the text encoder become more accurate. Therefore, to further prompt the pre-trained language model, we use the context of the given node's subgraph based on residual connection between text encoder and graph encoder. We first send the text embeddings of class label and node embeddings to the self-attention layer, in order to facilitate the text features to find the most relevant contextual nodes of the given node. Having the output of the attention layer De, we then update the text features via a residual connection as follows:

$$Text(P) \leftarrow Text(P) + \lambda De, \tag{14}$$

where λ is a parameter to control the scaling of the residual. We initialize λ as a small value 10^{-4} so that the language priors from the text features could be maximally maintained.

To optimize the text vectors, we conduct training to minimize the standard classification loss based on cross-entropy. The gradients can be back-propagated through the text encoder $Text(\cdot)$, in which the rich knowledge encoded in the parameters could be used. Our choice of continuous vectors also enables the full exploration of the word embedding space, which improves the learning of task-relevant text.

Summary. The pseudocode in Algorithm 1 summarizes the comprehensive learning workflow of P-HIN. Given a textual HIN, we first conduct the pre-training phase (line 1). We first generate the pre-trained representations for textual information (line 3). Then, we generate the pre-trained representations for HIN via the proposed modules, i.e., structure module, heterogeneous module, and self-supervised pre-training module (line 4–9). After that, we calculate the $\mathcal{L}_{Overall-loss}$ to align the textual data and graph data, generating the final pre-trained text encoder. Next, we move on to the prompt learning phase (line 13). Specifically, we first generate the continuous prompt (line 15) and add a residual connection to the text encoder (line 16). Finally, we calculate the prediction probability of the [CLASS] token to facilitate various downstream tasks.

5 Experimental Setup

In this section, we introduce the details of the real-world datasets, baseline models, and parameter settings.

5.1 Datasets

We consider four real-world datasets, i.e., OAG,[1] YELP,[2] Reddit,[3] and Amazon [20]. OAG is a bibliographic graph network having four types of nodes, i.e., author, paper, venue, and institute. We choose the title and abstract as texts. We extract a subset of it, covering five kinds of labels of the paper nodes and venue nodes: (i) database, (ii) data mining, (iii) machine learning, (iv) NLP, and (v) information retrieval. We choose two types of nodes to be classified to further illustrate that our model is able to process heterogeneous settings. These two types of nodes share the same classification labels, while they are trained separately.

YELP is a venue check-in network and has four types of nodes, i.e., review, customer, restaurant, and food-related keywords. The textual descriptions are the reviews of the restaurant, while the restaurants are divided into five kinds of labels: (i) Chinese food, (ii) Indian food, (iii) French food, (iv) fast food, and (v) sushi bar.

Reddit is a dataset extracted from the online forum Reddit (based on the posts made in the month of October 2022) and has three types of nodes, i.e., user, post, and comment; textual descriptions are titles and contents of posts. The label to be classified is the community a post belongs to. OAG and YELP only have 5 labels for few-shot classification, while Reddit has 41 labels, which could verify that P-HIN could fit into different practical scenarios.

We also include the datasets used in PATTON for a fair comparison, Amazon [20]. It is an e-commerce network and contains items about sports from Amazon. It is also an HIN as it has three types of nodes, i.e., user, item, and review. It has 16 labels for item nodes, classifying the specific sport the item belongs to. The description of the items is regarded as the textual information.

We split the datasets into 80% training datasets, 10% validation datasets and 10% testing datasets. Baselines are all trained based on this setting. Table 1 summarizes the information about the above datasets.

[1] https://www.microsoft.com/en-us/research/project/open-academic-graph.
[2] https://www.yelp.com/dataset_challenge.
[3] https://www.reddit.com.

Algorithm 1: The Learning Pseudocode of P-HIN

Input : Textual Heterogeneous Graphs G; Pre-train epoch κ

```
Pre-train phase;
while not done do
    Generate text pre-trained representations;
    for i=1 to k do
        Capture structure information via structural module;
        Capture heterogeneous information via heterogeneous module;
        Self-supervised pre-training;
    end
    Generate pre-trained node embeddings;
    Calculate L_Overall−loss via Eq. (10);
    Generate the pre-trained text encoder;
end
Prompt-learning phase;
while not done do
    Generate the continuous prompt;
    Add residual connection to the text encoder;
    Calculate prediction probability of [CLASS] token via Eq. (13);
end
```

Table 1. Dataset Statistics

Dataset	#nodes	#edges	#labels
OAG	432,362	1,837,362	5
YELP	382,472	2,412,428	5
Reddit	284,674	24,763,542	41
Amazon	314,448	3,461,379	16

5.2 Baselines

We first choose several baselines that learn textual network embeddings, that is, TADW [44], CENE [36], CANE [39], WANE [33], NEIFA [43], and DetGP [4]. We also include multi-modal models that apply a "pre-training and fine-tuning" framework, i.e., DRAGON [48] and PATTON [20]. Then, we choose HeCo [42] and PT-HGNN [17] as representative models of representation learning for HINs without using textual information. HeCo is a typical model using GNN encoders to encode HINs and PT-HGNN applies pre-training techniques for HIN modeling. In addition, we choose representation learning models that apply prompting techniques, that is, GPPT [34], GraphPrompt [28], GPF [7], HetGPT [29], EdgePrompt [12], and IA-GPL [23]. Sun et al. [35] did not assign a name for their proposed model; for convenience, we name their model AllinOne. GPPT is only applicable to node classification tasks. GraphPrompt is only applicable to node classification and graph classification tasks. GPF is only applicable to graph classification. AllinOne is applicable to node classification, graph classification, and link prediction. HetGPT is only applicable to node classification. EdgePrompt and IA-GPL are only applicable to node classification and graph classification.

5.3 Parameters

The latent dimensions of all representations are fixed to 512. For the text encoder, the vocabulary size is 49,152 and each text sequence is fixed to 77 with the [SOS] and [EOS] encompassed. The text vectors in optimization are initialized by a zero-mean Gaussian distribution with standard deviation equal to 0.02. The number of text tokens is set to 8. We adopt SGD for training with an initial learning rate of 0.002; it is decayed using the cosine annealing rule. The maximum epoch is set to 200. To mitigate explosive gradients observed in early training iterations, we use the warmup trick by fixing the learning rate to $1e-5$ during the first epoch. Three labels are used for training in OAG and YELP, and the rest for testing. Data with 31 labels is used for training on the Reddit dataset, with the rest used for testing. For the baselines, we directly adopt the best parameter configurations reported in their original papers. Baselines use the model trained on the training dataset to classify the unseen samples in their testing dataset. We use 1-shot, 3-shot, and 5-shot samples per class for training. Pre-training and prompt-tuning stages share the same experimental parameters.

We use an Intel(R) Xeon(R) Platinum 8268 CPU and Tesla V100 to run experiments with both the pre-training and downstream tasks.

6 Experimental Results

Our experiments cover multiple typical graph tasks, i.e., node-level (node classification), graph-level (graph classification), and edge-level (link prediction) tasks. In addition, we also introduce the keyword generation task to verify our model's broad application. We introduce the results of these tasks in the following subsections. In addition, we also include ablation analysis, parameter analysis, and efficiency evaluation to comprehensively analyze the proposed model.

6.1 Node Classification

We first evaluate the performance of P-HIN and the baselines on the node classification task. Note that OAG has two types of nodes to be classified; we use OAG-P and OAG-V to denote the results of classifying paper nodes and venue nodes, respectively. Accuracy (ACC) and Macro-F1 score are adopted as evaluation metrics (averaged over five folds). The experimental results for the node classification task under 1-shot, 3-shot, and 5-shot settings are displayed in Tables 2–4, respectively, with the highest scores emphasized in bold. "Improvement" denotes P-HIN's improvement compared with the best baseline PATTON. We performed a paired two-tailed t-test to determine statistical significance, with a marked improvement of P-HIN over the top-performing baseline PATTON (for $p < 0.05$) indicated by the symbol$^{\blacktriangle}$.

P-HIN consistently and significantly outperforms the baselines on all datasets under 1-shot, 3-shot, and 5-shot settings, which demonstrates the model's effectiveness. Specifically, all textual network embedding models underperform compared to P-HIN, due to their inability to address the few-shot learning problem. GPPT, GraphPrompt, AllinOne, HetGPT, EdgePrompt, and IA-GPL have comparable performance with DetGP even without using textual information. This is because their prompt technique helps to deal with few-shot settings. HeCo and PT-HGNN perform worse than graph prompt methods, further illustrating the effectiveness of prompt techniques. GRAGON and PATTON perform better than other baselines that pre-train the graph and language model to realize multi-modal learning. This verifies the power of encoding both text and graph information. However, this pre-training strategy leaves a gap between pre-training and downstream tasks. P-HIN performs the best, and with fewer training shots, the improvement gap increases, which further verifies P-HIN's ability to deal with few-shot settings. We believe its solid performance can be attributed to the fact that P-HIN provides a new angle to use textual information of graphs based

Table 2. Results on the 1-Shot Node Classification Task

	OAG-P		OAG-V		YELP		Reddit		Amazon	
Model	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1
TADW	0.211	0.279	0.232	0.303	0.188	0.256	0.078	0.196	0.146	0.221
CENE	0.217	0.288	0.236	0.312	0.195	0.261	0.084	0.202	0.157	0.228
CANE	0.228	0.297	0.251	0.336	0.201	0.268	0.087	0.210	0.162	0.234
WANE	0.233	0.301	0.257	0.343	0.207	0.274	0.094	0.218	0.171	0.239
NEIFA	0.241	0.307	0.265	0.351	0.215	0.278	0.099	0.224	0.177	0.246
DetGP	0.259	0.322	0.302	0.378	0.225	0.295	0.105	0.232	0.194	0.264
DRAGON	0.268	0.334	0.311	0.388	0.231	0.311	0.118	0.241	0.205	0.281
PATTON	0.272	0.341	0.316	0.394	0.238	0.318	0.126	0.244	0.218	0.288
Heco	0.236	0.296	0.261	0.341	0.195	0.257	0.078	0.202	0.165	0.238
PT-HGNN	0.242	0.302	0.266	0.347	0.203	0.268	0.085	0.214	0.172	0.244
GraphPrompt	0.257	0.320	0.283	0.357	0.223	0.292	0.103	0.230	0.192	0.260
GPPT	0.261	0.322	0.288	0.364	0.221	0.286	0.105	0.235	0.195	0.268
AllinOne	0.263	0.319	0.291	0.363	0.236	0.295	0.106	0.228	0.197	0.271
HetGPT	0.268	0.334	0.295	0.379	0.237	0.304	0.110	0.235	0.201	0.275
EdgePrompt	0.266	0.328	0.292	0.372	0.235	0.298	0.108	0.231	0.196	0.266
IA-GPL	0.264	0.329	0.287	0.369	0.229	0.296	0.104	0.229	0.190	0.261
P-HIN	**0.347**$^{\blacktriangle}$	**0.426**$^{\blacktriangle}$	**0.388**$^{\blacktriangle}$	**0.471**$^{\blacktriangle}$	**0.311**$^{\blacktriangle}$	**0.392**$^{\blacktriangle}$	**0.158**$^{\blacktriangle}$	**0.323**$^{\blacktriangle}$	**0.275**$^{\blacktriangle}$	**0.365**$^{\blacktriangle}$
Improvement	21.6%	20.0%	22.8%	19.5%	23.5%	18.9%	20.3%	24.5%	20.9%	21.1%

Table 3. Results on the 3-Shot Node Classification Task

	OAG-P		OAG-V		YELP		Reddit		Amazon	
Model	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1
TADW	0.404	0.512	0.432	0.547	0.395	0.481	0.295	0.398	0.331	0.453
CENE	0.418	0.520	0.445	0.554	0.401	0.490	0.304	0.403	0.338	0.458
CANE	0.423	0.525	0.452	0.558	0.408	0.496	0.311	0.412	0.342	0.464
WANE	0.429	0.528	0.454	0.566	0.418	0.501	0.321	0.421	0.346	0.467
NEIFA	0.438	0.536	0.461	0.572	0.426	0.506	0.327	0.427	0.351	0.471
DetGP	0.447	0.539	0.470	0.577	0.430	0.511	0.338	0.438	0.375	0.482
DRAGON	0.457	0.543	0.478	0.584	0.437	0.518	0.345	0.443	0.384	0.490
PATTON	0.461	0.546	0.484	0.591	0.441	0.523	0.349	0.448	0.388	0.495
HeCo	0.420	0.507	0.444	0.550	0.407	0.483	0.310	0.408	0.349	0.465
PT-HGNN	0.425	0.513	0.447	0.554	0.413	0.489	0.314	0.417	0.354	0.469
GraphPrompt	0.444	0.528	0.464	0.569	0.428	0.510	0.336	0.437	0.372	0.477
GPPT	0.441	0.530	0.460	0.565	0.425	0.508	0.334	0.432	0.373	0.480
AllinOne	0.450	0.539	0.472	0.575	0.428	0.506	0.338	0.440	0.378	0.484
HetGPT	0.455	0.535	0.474	0.572	0.435	0.511	0.340	0.442	0.382	0.487
EdgePrompt	0.453	0.532	0.473	0.570	0.432	0.507	0.337	0.441	0.377	0.486
IA-GPL	0.448	0.531	0.469	0.570	0.425	0.504	0.337	0.437	0.374	0.480
P-HIN	**0.517**$^{\blacktriangle}$	**0.627**$^{\blacktriangle}$	**0.530**$^{\blacktriangle}$	**0.641**	**0.505**$^{\blacktriangle}$	**0.577**$^{\blacktriangle}$	**0.401**$^{\blacktriangle}$	**0.512**$^{\blacktriangle}$	**0.459**$^{\blacktriangle}$	**0.553**$^{\blacktriangle}$
Improvement	10.8%	12.9%	9.5%	8.5%	12.7%	9.4%	13.0%	12.5%	15.4%	10.5%

on prompt learning, facilitating multi-modal learning and meanwhile helping P-HIN to fit in the few-shot setting. It also narrows down the gap between the pre-training and downstream tasks. In addition, previous methods are not specifically designed for heterogeneous networks, while P-HIN comes with a graph encoder that can handle heterogeneous features.

Table 4. Results on the 5-Shot Node Classification Task

	OAG-P		OAG-V		YELP		Reddit		Amazon	
Model	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1
TADW	0.541	0.638	0.562	0.660	0.521	0.576	0.459	0.523	0.472	0.542
CENE	0.554	0.644	0.574	0.667	0.534	0.584	0.468	0.539	0.479	0.546
CANE	0.565	0.653	0.580	0.676	0.547	0.601	0.475	0.554	0.482	0.558
WANE	0.584	0.686	0.602	0.694	0.563	0.626	0.481	0.567	0.488	0.572
NEIFA	0.604	0.706	0.619	0.719	0.574	0.658	0.489	0.579	0.494	0.584
DetGP	0.612	0.711	0.631	0.734	0.584	0.667	0.496	0.591	0.518	0.615
DRAGON	0.618	0.717	0.638	0.739	0.590	0.672	0.502	0.597	0.526	0.624
PATTON	0.621	0.721	0.640	0.744	0.595	0.676	0.506	0.603	0.538	0.639
HeCo	0.583	0.677	0.604	0.692	0.551	0.604	0.443	0.539	0.461	0.575
PT-HGNN	0.589	0.683	0.610	0.697	0.558	0.621	0.458	0.558	0.486	0.593
GraphPrompt	0.606	0.709	0.621	0.723	0.582	0.666	0.493	0.584	0.513	0.612
GPPT	0.603	0.707	0.619	0.721	0.579	0.663	0.487	0.575	0.517	0.612
AllinOne	0.609	0.711	0.625	0.727	0.589	0.668	0.496	0.589	0.520	0.617
HetGPT	0.615	0.713	0.623	0.725	0.591	0.672	0.501	0.593	0.523	0.621
EdgePrompt	0.612	0.710	0.618	0.716	0.586	0.665	0.499	0.595	0.519	0.618
IA-GPL	0.604	0.705	0.616	0.715	0.583	0.657	0.488	0.578	0.511	0.609
P-HIN	**0.652**▲	**0.749**▲	**0.675**▲	**0.775**▲	**0.624**▲	**0.716**▲	**0.535**▲	**0.644**▲	**0.578**▲	**0.674**▲
Improvement	4.8%	3.7%	5.5%	4.2%	4.6%	5.6%	5.4%	6.4%	6.9%	5.2%

6.2 Graph Classification

Next, we evaluate the performance of P-HIN and the baselines on the graph classification task. Accuracy (ACC) and Macro-F1 score are adopted as evaluation metrics (averaged over five folds). The experimental results for the graph classification task under 1-shot, 3-shot, and 5-shot settings are displayed in Tables 5–7, respectively, with the highest scores emphasized in bold. "Improvement" denotes P-HIN's improvement compared with the best baseline PATTON. We performed a paired two-tailed t-test to determine statistical significance, with a marked improvement of P-HIN over the top-performing baseline PATTON (for $p < 0.05$) indicated by the symbol▲.

Similar to the experimental results on the node classification task, P-HIN achieves the best experimental results, and with fewer shots, P-HIN's advantage increases. It shows P-HIN's ability to process few-shot classification problem. This can be attributed to the prompt technique we use, which aligns textual information and graph information, while narrowing the gap between pre-training and downstream tasks.

6.3 Link Prediction

Next, we adjust the fine-tuning stage to enable the model to realize the link prediction task. Specifically, we transfer the link prediction task into a binary classification task, where the label indicates if a node pair in a subgraph is connected or not. We adopt the AUC value and Macro-F1 value as evaluation metrics (five-fold average).

The experimental results for the link prediction task under 1-shot, 3-shot, and 5-shot settings are displayed in Tables 8–10, respectively, with the highest scores emphasized in bold. "Improvement" denotes P-HIN's improvement compared with the best baseline PATTON. We performed a paired two-tailed t-test to determine statistical significance, with a marked improvement of P-HIN over the top-performing baseline PATTON (for $p< 0.05$) indicated by the symbol▲.

Table 5. Results on the 1-Shot Graph Classification Task

	OAG		YELP		Reddit		Amazon	
Model	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1
TADW	0.398	0.472	0.357	0.413	0.265	0.402	0.278	0.423
CENE	0.416	0.477	0.396	0.421	0.291	0.411	0.304	0.435
CANE	0.429	0.502	0.409	0.472	0.297	0.418	0.309	0.443
WANE	0.437	0.507	0.411	0.487	0.302	0.421	0.315	0.454
NEIFA	0.445	0.511	0.421	0.479	0.298	0.419	0.321	0.460
DetGP	0.465	0.527	0.432	0.501	0.309	0.443	0.338	0.469
DRAGON	0.482	0.542	0.451	0.523	0.345	0.467	0.369	0.482
PATTON	0.485	0.548	0.456	0.529	0.352	0.474	0.377	0.489
HeCo	0.438	0.495	0.407	0.472	0.285	0.416	0.306	0.434
PT-HGNN	0.444	0.504	0.414	0.483	0.293	0.425	0.317	0.446
GraphPrompt	0.462	0.528	0.431	0.496	0.311	0.441	0.333	0.465
GPF	0.463	0.526	0.428	0.499	0.308	0.438	0.339	0.464
AllinOne	0.466	0.531	0.427	0.494	0.312	0.436	0.341	0.468
EdgePrompt	0.464	0.530	0.423	0.490	0.314	0.439	0.336	0.463
IA-GPL	0.460	0.522	0.421	0.488	0.305	0.428	0.330	0.463
P-HIN	**0.558**▲	**0.631**▲	**0.524**▲	**0.599**▲	**0.398**▲	**0.541**▲	**0.425**▲	**0.540**▲
Improvement	13.1%	13.2%	13.0%	11.7%	11.6%	12.4%	11.3%	9.4%

Table 6. Results on the 3-Shot Graph Classification Task

	OAG		YELP		Reddit		Amazon	
Model	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1
TADW	0.612	0.709	0.598	0.684	0.496	0.611	0.547	0.634
CENE	0.621	0.723	0.603	0.691	0.502	0.612	0.556	0.645
CANE	0.633	0.712	0.604	0.694	0.504	0.614	0.561	0.649
WANE	0.635	0.732	0.612	0.703	0.502	0.621	0.567	0.653
NEIFA	0.639	0.738	0.615	0.701	0.505	0.626	0.571	0.658
DetGP	0.645	0.741	0.618	0.705	0.511	0.634	0.578	0.664
DRAGON	0.661	0.749	0.629	0.711	0.529	0.642	0.588	0.675
PATTON	0.665	0.753	0.632	0.715	0.532	0.647	0.594	0.681
HeCo	0.619	0.715	0.593	0.667	0.477	0.587	0.523	0.612
PT-HGNN	0.624	0.721	0.603	0.682	0.489	0.605	0.549	0.637
GraphPrompt	0.647	0.739	0.619	0.704	0.517	0.632	0.575	0.662
GPF	0.649	0.732	0.621	0.707	0.521	0.631	0.577	0.665
AllinOne	0.646	0.737	0.618	0.705	0.523	0.629	0.581	0.670
EdgePrompt	0.651	0.741	0.625	0.710	0.527	0.635	0.579	0.667
IA-GPL	0.644	0.729	0.612	0.699	0.511	0.625	0.572	0.660
P-HIN	**0.717**▲	**0.811**▲	**0.698**▲	**0.779**▲	**0.611**▲	**0.709**▲	**0.646**▲	**0.733**▲
Improvement	7.3%	7.2%	9.5%	8.2%	12.9%	8.7%	8.0%	7.2%

P-HIN outperforms all the baselines under all training shot settings, and with a smaller number of training shots, P-HIN's advantage increases, which further proves the proposed model's ability to handle the few-shot link prediction task. This can be attributed to two main reasons. One is that

Table 7. Results on the 5-Shot Graph Classification Task

	OAG		YELP		Reddit		Amazon	
Model	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1
TADW	0.734	0.801	0.701	0.752	0.638	0.701	0.661	0.724
CENE	0.738	0.804	0.706	0.756	0.641	0.705	0.665	0.728
CANE	0.742	0.805	0.703	0.757	0.639	0.703	0.671	0.732
WANE	0.749	0.807	0.710	0.761	0.643	0.707	0.674	0.738
NEIFA	0.756	0.811	0.717	0.767	0.649	0.713	0.680	0.743
DetGP	0.769	0.821	0.729	0.779	0.662	0.729	0.685	0.748
DRAGON	0.774	0.831	0.738	0.786	0.667	0.735	0.691	0.755
PATTON	0.778	0.836	0.744	0.791	0.674	0.741	0.696	0.761
HeCo	0.739	0.782	0.681	0.727	0.616	0.695	0.655	0.711
PT-HGNN	0.743	0.793	0.695	0.743	0.630	0.703	0.661	0.715
GraphPrompt	0.765	0.818	0.731	0.776	0.652	0.725	0.680	0.741
GPF	0.769	0.821	0.734	0.779	0.653	0.727	0.683	0.745
AllinOne	0.768	0.823	0.737	0.782	0.657	0.730	0.688	0.752
EdgePrompt	0.771	0.825	0.739	0.785	0.660	0.732	0.685	0.747
IA-GPL	0.765	0.816	0.726	0.775	0.647	0.720	0.677	0.740
P-HIN	**0.818**▲	**0.878**▲	**0.789**▲	**0.819**▲	**0.709**▲	**0.779**▲	**0.741**▲	**0.804**▲
Improvement	4.9%	4.8%	5.7%	3.4%	4.9%	4.9%	6.0%	5.3%

Table 8. Results on the 1-Shot Link Prediction Task

	OAG		YELP		Reddit		Amazon	
Model	AUC	MAC-F1	AUC	MAC-F1	AUC	MAC-F1	AUC	MAC-F1
TADW	0.298	0.256	0.301	0.253	0.201	0.109	0.251	0.167
CENE	0.302	0.265	0.307	0.268	0.210	0.121	0.257	0.173
CANE	0.313	0.272	0.312	0.275	0.217	0.131	0.263	0.178
WANE	0.319	0.281	0.326	0.282	0.226	0.140	0.267	0.184
NEIFA	0.326	0.287	0.336	0.287	0.231	0.147	0.270	0.190
DetGP	0.332	0.293	0.354	0.296	0.242	0.154	0.277	0.196
GRAGON	0.328	0.286	0.341	0.265	0.232	0.142	0.241	0.169
PATTON	0.332	0.294	0.356	0.287	0.248	0.159	0.258	0.186
HeCo	0.332	0.293	0.354	0.296	0.242	0.154	0.277	0.196
PT-HGNN	0.332	0.293	0.354	0.296	0.242	0.154	0.277	0.196
AllinOne	0.329	0.294	0.357	0.291	0.245	0.152	0.276	0.198
P-HIN	**0.425**▲	**0.386**▲	**0.449**▲	**0.397**▲	**0.332**▲	**0.230**▲	**0.357**▲	**0.256**▲
Improvement	17.2%	18.9%	20.2%	16.1%	18.7%	20.4%	19.9%	19.5%

we adopt the ER pre-training task which already explores the edge-level structure, boosting the link prediction performance. The other is similar to what we illustrated in Section 6.1: the prompt learning setting helps to effectively mine the text features, leading to a better performance on downstream tasks.

Table 9. Results on the 3-Shot Link Prediction Task

	OAG		YELP		Reddit		Amazon	
Model	AUC	MAC-F1	AUC	MAC-F1	AUC	MAC-F1	AUC	MAC-F1
TADW	0.532	0.491	0.601	0.477	0.583	0.476	0.589	0.486
CENE	0.583	0.495	0.610	0.482	0.594	0.485	0.593	0.494
CANE	0.586	0.504	0.617	0.487	0.603	0.494	0.598	0.497
WANE	0.591	0.513	0.623	0.494	0.612	0.501	0.608	0.500
NEIFA	0.595	0.517	0.630	0.502	0.618	0.503	0.611	0.505
DetGP	0.603	0.526	0.638	0.512	0.627	0.511	0.626	0.518
GRAGON	0.611	0.538	0.653	0.527	0.635	0.519	0.639	0.525
PATTON	0.617	0.542	0.659	0.531	0.638	0.524	0.645	0.529
HeCo	0.567	0.487	0.594	0.463	0.589	0.465	0.579	0.477
PT-HGNN	0.579	0.501	0.605	0.479	0.603	0.483	0.606	0.499
AllinOne	0.601	0.524	0.633	0.506	0.627	0.512	0.623	0.517
P-HIN	**0.727**▲	**0.613**▲	**0.737**▲	**0.624**▲	**0.731**▲	**0.619**▲	**0.721**▲	**0.615**▲
Improvement	15.1%	11.6%	10.6%	14.9%	12.7%	15.3%	10.5%	14.1%

Table 10. Results on the 5-Shot Link Prediction Task

	OAG		YELP		Reddit		Amazon	
Model	AUC	MAC-F1	AUC	MAC-F1	AUC	MAC-F1	AUC	MAC-F1
TADW	0.724	0.635	0.773	0.642	0.791	0.704	0.785	0.667
CENE	0.745	0.641	0.794	0.664	0.805	0.736	0.793	0.682
CANE	0.773	0.667	0.810	0.685	0.823	0.753	0.800	0.692
WANE	0.792	0.683	0.833	0.694	0.845	0.768	0.816	0.712
NEIFA	0.826	0.694	0.856	0.743	0.866	0.783	0.827	0.728
DetGP	0.843	0.732	0.877	0.788	0.871	0.793	0.836	0.736
GRAGON	0.852	0.745	0.884	0.795	0.876	0.802	0.845	0.747
PATTON	0.858	0.751	0.888	0.780	0.882	0.807	0.859	0.761
HeCo	0.803	0.685	0.823	0.729	0.806	0.749	0.789	0.678
PT-HGNN	0.812	0.704	0.849	0.751	0.829	0.766	0.807	0.701
AllinOne	0.847	0.738	0.874	0.784	0.873	0.792	0.840	0.741
P-HIN	**0.897**▲	**0.792**▲	**0.912**▲	**0.823**▲	**0.901**▲	**0.837**▲	**0.887**▲	**0.801**▲
Improvement	4.3%	5.2%	2.6%	5.2%	2.1%	3.6%	3.2%	5.0%

6.4 Keyword Generation

Here, we introduce a task that has not been considered in the graph evaluation domain, i.e., keyword generation. P-HIN is able to generate keywords because of its autoregressive generation abilities under the pre-training and prompt learning framework. In practice, we use the keywords of a paper node in OAG as ground-truth labels. When fine-tuning, we have the text [CLS][MASK], with [CLS] being the first word of the keywords and next work to be predicted in the [MASK] token. This process will stop when [SEP] is returned. The traditional textual network embedding methods are not able to generate keywords. So to enable a comparison, we use KeyBERT [13], which generates keywords based on BERT-embeddings. It takes the description of each node as input. We adopt the F1 measure on the top 1 and top 3 keyword predictions as our evaluation metric.

Table 11. Results on the Keyword Generation Task

	OAG	
Model	F1@1	F1@3
KeyBERT	0.287	0.344
P-HIN	**0.342**▲	**0.401**▲

The bold denotes the highest score.

Table 11 presents the results for the keyword generation task. P-HIN outperforms KeyBERT as P-HIN uses both textual information and subgraph information, while KeyBERT is only able to use the textual information.

6.5 Ablation Study

To evaluate the effectiveness of each component or strategy the model uses, we perform an ablation study. It is conducted on the 5-shot node classification, graph classification, and link prediction tasks.

Recall that P-HIN adopts a pre-training and prompt tuning framework; below, we first evaluate the effects of pre-training and prompt tuning, respectively. P-HIN\prompt directly uses the pre-trained graph encoder to conduct the downstream tasks. P-HIN\pretrain directly uses the graph encoder and text encoder without pre-training to conduct the downstream tasks.

To assess the individual contributions of the encoders, we introduce two ablated variants: P-HIN\Graph, which relies solely on the text encoder for downstream tasks, and P-HIN\Text, which uses only the graph encoder.

To evaluate the effectiveness of the text encoder SBERT, we introduce two other commonly used text encoders, BERT and XLNet, denoted as P-HIN-BERT and P-HIN-XLNet.

For the text encoder, we adopt RWR as the heterogeneous subgraph sampling strategy. To evaluate the chosen strategy, we introduce three variants which use random walks, **Breadth-First Search (BFS)** and **Depth-First Search (DFS)** to sample the subgraphs, which are denoted as P-HIN-RandomWalk, P-HIN-BFS, and P-HIN-DFS, respectively.

To evaluate the continuous vectors we adopt during prompt tuning, we introduce a variant denoted as P-HIN-Manual that manually creates the prompts. The prompts are designed as "a paper of [CLASS] domain" in OAG, "a restaurant of [CLASS] label" in YELP and "a post belongs to [CLASS] community" in Reddit.

During text-subgraph matching pre-training, we limit the negative samples to be sampled from the same label so that they will be harder to be distinguished. To evaluate this strategy, we introduce a variant that randomly forms the negative samples, denoted as P-HIN-Random.

During pre-training the graph, we introduce two self-supervision tasks, i.e., MNM and ER. To evaluate the effectiveness of these tasks, we introduce three variants denoted as P-HIN\MNM, P-HIN\ER, and P-HIN\Self. P-HIN\MNM excludes the MNM task, P-HIN\ER excludes the ER task and P-HIN\Self excludes both self-supervision tasks.

We also adopt the residual connection between the text encoder and graph encoder to assign the given node's context sense to the text feature. To evaluate this module, we introduce a variant denoted as P-HIN\Residual that ignores the residual connection.

Tables 12–14 present the results of the ablation study. We make the following observations.

Table 12. Results of Our Ablation Study on the Node Classification Task

	OAG		YELP		Reddit		Amazon	
Model	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1
P-HIN\Prompt	0.578	0.618	0.553	0.604	0.435	0.547	0.478	0.577
P-HIN\Pretrain	0.553	0.602	0.528	0.586	0.421	0.526	0.457	0.549
P-HIN\Graph	0.364	0.498	0.331	0.477	0.295	0.315	0.321	0.403
P-HIN\Text	0.607	0.704	0.587	0.667	0.482	0.580	0.523	0.625
P-HIN-BERT	0.623	0.712	0.599	0.683	0.501	0.602	0.534	0.638
P-HIN-XLNet	0.628	0.719	0.605	0.693	0.505	0.611	0.540	0.642
P-HIN-RandomWalk	0.612	0.721	0.592	0.677	0.496	0.594	0.532	0.633
P-HIN-BFS	0.604	0.715	0.583	0.666	0.486	0.578	0.525	0.628
P-HIN-DFS	0.606	0.718	0.589	0.669	0.491	0.584	0.522	0.626
P-HIN-Manual	0.641	0.732	0.608	0.692	0.518	0.622	0.554	0.647
P-HIN-Random	0.637	0.735	0.617	0.701	0.521	0.628	0.559	0.652
P-HIN\MNM	0.626	0.712	0.606	0.689	0.506	0.610	0.542	0.641
P-HIN\ER	0.633	0.716	0.617	0.694	0.511	0.617	0.544	0.644
P-HIN\Self	0.610	0.703	0.594	0.673	0.498	0.597	0.536	0.637
P-HIN\Residual	0.636	0.733	0.611	0.695	0.517	0.619	0.548	0.649
P-HIN	**0.652**▲	**0.749**▲	**0.624**▲	**0.716**▲	**0.535**▲	**0.644**▲	**0.578**▲	**0.674**▲

The bold denotes the highest score.

The performance of P-HIN\Prompt and P-HIN\Pretrain drops significantly compared with the proposed P-HIN, illustrating the effectiveness of the pre-training and prompt tuning framework as a whole.

P-HIN\Graph and P-HIN\Text all have much worse performance than P-HIN, which proves the effectiveness of the proposed model's choice by integrating multi-modal information, i.e., textual and graph information. P-HIN\Graph performs the worst, which demonstrates that graph encoder contributes more on mining textual HIN by exploring heterogeneous and structural information.

P-HIN surpasses both P-HIN-BERT and P-HIN-XLNet, demonstrating that SBERT extracts more comprehensive textual information than BERT and XLNet.

P-HIN outperforms P-HIN-RandomWalk, P-HIN-BFS, and P-HIN-DFS, which verifies that RWR sampling strategy could better extract and maintain the features of the heterogeneous subgraphs.

P-HIN outperforms P-HIN-Manual, which uses manually crafted prompts. Learning continuous prompts is more flexible and caters to the specific task, which helps to boost the model performance.

P-HIN also outperforms P-HIN-Random, which randomly chooses negative samples to be compared. It can be verified that comparing high-quality negative samples could help to train the contrastive learning framework, thus improving the final experimental results.

The self-supervision tasks do help to improve the model performance. When excluding either of the tasks, the performance drops, with P-HIN\Self performing the worst. Notice that P-HIN\MNM performs worse than P-HIN\ER. MNM explores node-level information and ER explores edge-level information. The node classification task is more related to the MNM task.

P-HIN outperforms P-HIN\Residual, which confirms that considering the contextual information from the graph to prompt the language model will help to improve the performance.

Table 13. Results of Our Ablation Study on the Graph Classification Task

	OAG		YELP		Reddit		Amazon	
Model	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1	ACC	MAC-F1
P-HIN\Prompt	0.613	0.665	0.538	0.561	0.442	0.502	0.493	0.511
P-HIN\Pretrain	0.602	0.642	0.509	0.534	0.421	0.488	0.474	0.495
P-HIN\Graph	0.534	0.583	0.489	0.499	0.392	0.446	0.443	0.465
P-HIN\Text	0.689	0.723	0.662	0.684	0.583	0.645	0.632	0.672
P-HIN-BERT	0.784	0.847	0.742	0.792	0.654	0.744	0.708	0.766
P-HIN-XLNet	0.788	0.849	0.746	0.801	0.661	0.748	0.711	0.772
P-HIN-RandomWalk	0.762	0.831	0.721	0.782	0.632	0.723	0.685	0.744
P-HIN-BFS	0.764	0.835	0.724	0.785	0.638	0.729	0.687	0.750
P-HIN-DFS	0.765	0.832	0.725	0.783	0.635	0.727	0.688	0.747
P-HIN-Manual	0.793	0.862	0.771	0.801	0.688	0.760	0.728	0.789
P-HIN-Random	0.789	0.853	0.762	0.799	0.667	0.749	0.717	0.775
P-HIN\MNM	0.772	0.846	0.735	0.785	0.645	0.743	0.701	0.761
P-HIN\ER	0.776	0.850	0.741	0.787	0.649	0.751	0.704	0.772
P-HIN\Self	0.752	0.831	0.722	0.763	0.621	0.725	0.684	0.740
P-HIN\Residual	0.778	0.849	0.751	0.793	0.637	0.739	0.707	0.767
P-HIN	**0.818**$^{\blacktriangle}$	**0.878**$^{\blacktriangle}$	**0.789**$^{\blacktriangle}$	**0.819**$^{\blacktriangle}$	**0.709**$^{\blacktriangle}$	**0.779**$^{\blacktriangle}$	**0.741**$^{\blacktriangle}$	**0.804**$^{\blacktriangle}$

The bold denotes the highest score.

6.6 Parameter Analysis

Here, we conduct a parameter analysis of P-HIN. We choose four parameters to be analyzed, i.e., (i) the number of the prompt tokens, (ii) the number of shots used for training, (iii) the percentage of masked nodes during MNM pre-training task, and (iv) the value of lambda in the residual connection. We conduct our parameter analysis on the node classification, graph classification, and link prediction tasks. And we adopt ACC for node and graph classification and AUC for link prediction as the evaluation metrics. Figures 3–5 present the experimental results of the parameter analyses.

Figure 3 shows the effect of the number of prompt tokens under 5-shot training setting. As that number increases, the performance improves correspondingly. We set the number to 8 in practice to balance effectiveness and efficiency. One can choose a higher number of text tokens for a better model performance.

Figure 4 shows the influence of the training shots. It demonstrates that P-HIN obtains a better performance when the number of shots increases as more shots will bring valuable information for training better model.

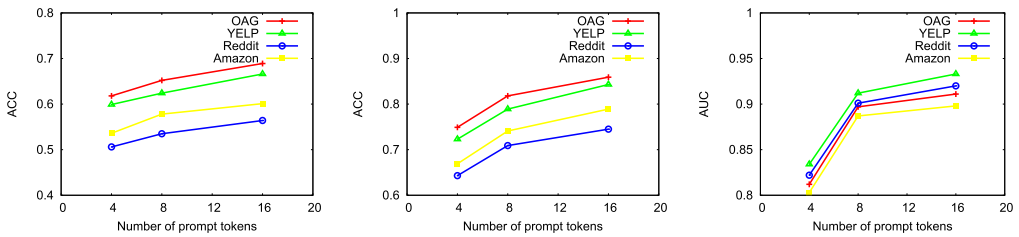
Figure 5 shows the effect of the percentage of masked nodes during MNM pre-training task under the 5-shot training setting. It illustrates that setting the masked percentage to 15% will obtain the best performance, which verifies our parameter choice.

Figure 6 shows the effect of the value of lambda in residual connection. It illustrates that when setting the value larger, that is, 10^{-2} and 10^{-3}, the results drops significantly. As lambda decreases, the results become more stable, with the best experimental result achieved when lambda is set to 10^{-4}.

Table 14. Results of Our Ablation Study on the Link Prediction Task

	OAG		YELP		Reddit		Amazon	
Model	AUC	MAC-F1	AUC	MAC-F1	AUC	MAC-F1	AUC	MAC-F1
P-HIN\Prompt	0.653	0.538	0.662	0.548	0.671	0.569	0.624	0.561
P-HIN\Pretrain	0.638	0.513	0.639	0.521	0.634	0.547	0.602	0.516
P-HIN\Graph	0.585	0.479	0.594	0.493	0.589	0.502	0.575	0.487
P-HIN\Text	0.691	0.585	0.702	0.597	0.695	0.611	0.673	0.592
P-HIN-BERT	0.868	0.770	0.872	0.795	0.870	0.801	0.852	0.765
P-HIN-XLNet	0.872	0.774	0.875	0.801	0.873	0.806	0.857	0.778
P-HIN-RandomWalk	0.842	0.749	0.848	0.774	0.852	0.779	0.832	0.748
P-HIN-BFS	0.848	0.754	0.856	0.781	0.862	0.785	0.839	0.759
P-HIN-DFS	0.844	0.756	0.849	0.782	0.858	0.782	0.834	0.752
P-HIN-Manual	0.874	0.772	0.889	0.803	0.783	0.812	0.874	0.787
P-HIN-Random	0.864	0.765	0.872	0.789	0.769	0.793	0.862	0.771
P-HIN\MNM	0.856	0.753	0.862	0.776	0.759	0.782	0.851	0.762
P-HIN\ER	0.842	0.741	0.853	0.769	0.751	0.773	0.838	0.749
P-HIN\Self	0.824	0.716	0.832	0.744	0.732	0.751	0.819	0.731
P-HIN\Residual	0.858	0.767	0.878	0.795	0.772	0.805	0.867	0.779
P-HIN	**0.897**▲	**0.792**▲	**0.912**▲	**0.823**▲	**0.901**▲	**0.837**▲	**0.887**▲	**0.801**▲

The bold denotes the highest score.



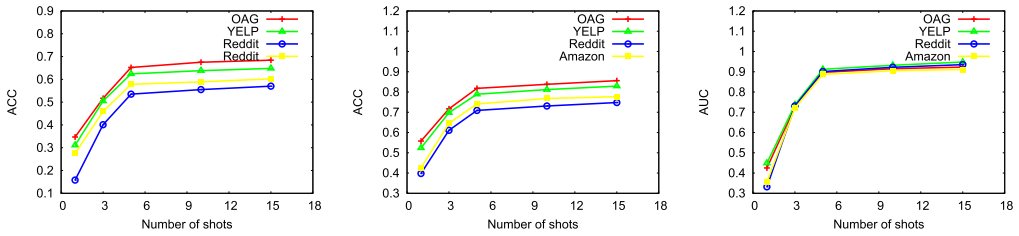
(a) Node classification (b) Graph classification (c) Link prediction

Fig. 3. Sensitivity with regard to number of prompt tokens.

6.7 Efficiency Evaluation

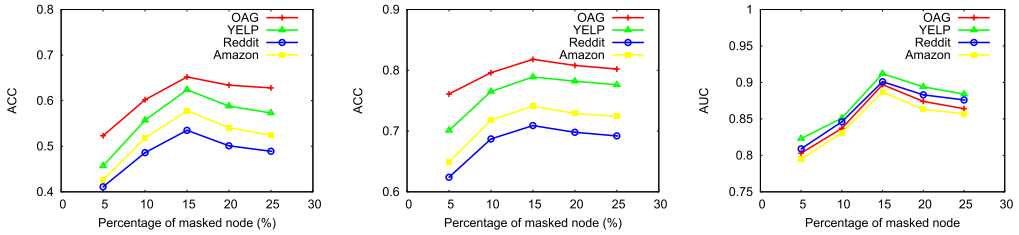
Here, we evaluate the model efficiency by comparing P-HIN's training time and inference time with other baselines on the node classification task. Table 15 presents the efficiency results.

GPPT, GraphPrompt, AllinOne, and P-HIN have comparable training and inference time, and they are more efficient than traditional textual network embedding methods with the help of pre-training and prompt framework. P-HIN has a longer running time than GraphPrompt as we need to sample heterogeneous subgraphs while GraphPrompt randomly samples subgraphs. Nevertheless, P-HIN outperforms GraphPrompt and is able to facilitate more downstream tasks. We believe that this efficiency sacrifice is acceptable.



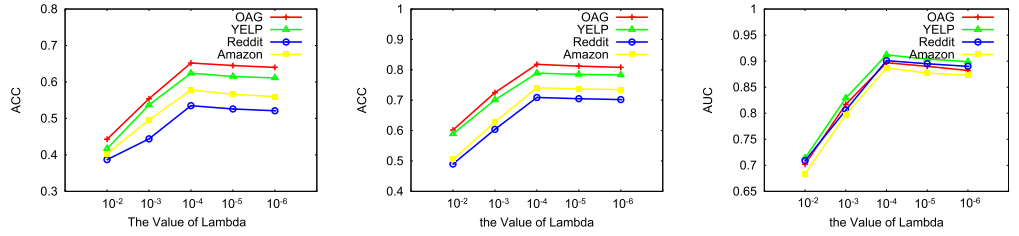
(a) Node classification (b) Graph Classification (c) Link prediction

Fig. 4. Sensitivity with regard to number of shots for training.



(a) Node classification (b) Graph classification (c) Link prediction

Fig. 5. Sensitivity with regard to percentage of masked node for training.



(a) Node Classification (b) Graph classification (c) Link prediction

Fig. 6. Sensitivity with regard to the value of lambda.

7 Conclusion and Limitations

Textual HINs play a crucial role in a wide range of real-world applications. This article presents a novel prompt learning framework P-HIN, which fuses textual information and graph information and handles the few-shot problems simultaneously. It is composed of a text encoder and a graph encoder. The generated textual embeddings and node embeddings are fused by a contrastive learning mechanism. We introduce MNM and ER tasks to make use of node-level and edge-level self-supervised information. We choose learnable continuous texts instead of manually designed prompts to facilitate the transfer of the pre-trained model. A residual connection is introduced to use the contextual information in graph to prompt the text model.

In our experiments, P-HIN consistently and significantly outperforms state-of-the-art methods on all datasets and on all downstream tasks, which demonstrates the advantages of P-HIN. More broadly, with P-HIN we now have a powerful few-shot learner for textual HINs.

In future work, we intend to investigate temporal evolution in HINs by incorporating time-aware mechanisms (e.g., temporal attention, sequence models) to capture the dynamics of node/edge

Table 15. Results of the Efficiency Evaluation (s/Epoch)

	OAG		YELP		Reddit		Amazon	
Model	Train	Inference	Train	Inference	Train	Inference	Train	Inference
TADW	421.4	15.8	398.2	14.5	378.8	14.2	386.2	15.1
CENE	403.2	15.3	371.3	13.8	355.2	13.6	360.4	13.9
CANE	413.3	16.1	389.4	14.7	366.4	13.9	382.4	14.6
WANE	389.6	12.7	367.1	12.3	326.9	11.9	343.8	12.6
NEIFA	383.2	11.4	378.6	11.6	352.4	11.5	370.3	11.8
DetGP	396.6	11.9	385.2	11.8	361.2	11.1	378.7	11.7
GPPT	150.3	3.5	141.2	3.1	133.8	2.7	138.5	2.9
GraphPrompt	138.4	3.2	120.5	2.4	110.4	2.1	113.6	2.2
AllinOne	152.4	3.8	139.5	3.4	121.5	2.9	128.8	3.6
P-HIN	148.5	3.3	137.3	2.7	124.6	2.5	132.9	2.7

changes, enable incremental learning for real-time updates. We also aim to integrate image and other modalities into a unified framework, and treating visual elements (objects, scenes) as distinct node types to enable fine-grained multimodal reasoning. The enhanced dynamic and multimodal framework can be further applied to high-impact domains such as multi-platform misinformation tracking, e-commerce recommendation with visual and behavioral cues, and scientific literature mining that combines text, figures, and datasets.

References

[1] Phillip Bonacich. 1987. Power and centrality: A family of measures. *American Journal of Sociology* 92, 5 (1987), 1170–1182.

[2] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.

[3] Chen Chen, Bin Song, Jie Guo, and Tong Zhang. 2023. Multi-dimensional shared representation learning with graph fusion network for session-based recommendation. *Information Fusion* 92 (2023), 205–215.

[4] Pengyu Cheng, Yitong Li, Xinyuan Zhang, Liqun Chen, David E. Carlson, and Lawrence Carin. 2020. Dynamic embedding on textual networks via a Gaussian process. In *AAAI*. AAAI Press, 7562–7569.

[5] Kaize Ding, Jundong Li, and Huan Liu. 2019. Interactive anomaly detection on attributed networks. In *WSDM*. ACM, 357–365.

[6] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, 135–144.

[7] Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. 2023. Universal prompt tuning for graph neural networks. In *NeurIPS*.

[8] Yang Fang, Xiang Zhao, Yifan Chen, Weidong Xiao, and Maarten de Rijke. 2023. PF-HIN:Pre-training for heterogeneous information networks. *IEEE Transactions on Knowledge and Data Engineering* 35, 8 (2023), 8372–8385.

[9] Yang Fang, Xiang Zhao, Peixin Huang, Weidong Xiao, and Maarten de Rijke. 2019. M-HIN: Complex embeddings for heterogeneous information networks via metagraphs. In *SIGIR*, 913–916.

[10] Yang Fang, Xiang Zhao, Peixin Huang, Weidong Xiao, and Maarten de Rijke. 2022. Scalable representation learning for dynamic heterogeneous information networks via metagraphs. *ACM Transactions on Information System* 40, 4, Article 64 (2022), 1–27.

[11] Tao-Yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. HIN2Vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM*, 1797–1806.

[12] Xingbo Fu, Yinhan He, and Jundong Li. 2025. Edge prompt tuning for graph neural networks. In *ICLR*. OpenReview.net.

[13] Maarten Grootendorst. 2020. KeyBERT: Minimal keyword extraction with BERT. Zenodo. Retrieved from https://doi.org/10.5281/zenodo.4461265

[14] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. arXiv:2003.00911. Retrieved from https://arxiv.org/abs/2003.00911

[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW*, 2704–2710.
[16] Zhipeng Huang and Nikos Mamoulis. 2017. Heterogeneous information network embedding for meta path based proximity. arXiv:1701.05291. Retrieved from https://arxiv.org/abs/1701.05291
[17] Xunqiang Jiang, Tianrui Jia, Yuan Fang, Chuan Shi, Zhe Lin, and Hui Wang. 2021. Pre-training on large-scale heterogeneous graph. In *KDD*. ACM, 756–766.
[18] Xunqiang Jiang, Yuanfu Lu, Yuan Fang, and Chuan Shi. 2021. Contrastive pre-training of GNNs on heterogeneous graphs. In *CIKM*. ACM, 803–812.
[19] Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Transactions of the Association for Computational Linguistics* 8 (2020), 423–438.
[20] Bowen Jin, Wentao Zhang, Yu Zhang, Yu Meng, Xinyang Zhang, Qi Zhu, and Jiawei Han. 2023. Patton: Language model pretraining on text-rich networks. In *ACL*. Association for Computational Linguistics, 7005–7020.
[21] Jiarui Jin, Kounianhua Du, Weinan Zhang, Jiarui Qin, Yuchen Fang, Yong Yu, Zheng Zhang, and Alexander J. Smola. 2022. GraphHINGE: Learning interaction models of structured neighborhood on heterogeneous information network. *ACM Transactions on Information System* 40, 3, Article 45 (2022), 1–35.
[22] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, 3045–3059.
[23] Jiazheng Li, Jundong Li, and Chuxu Zhang. 2025. Instance-aware graph prompt learning. *Transactions on Machine Learning Research* 25, 2 (2025), 1–20.
[24] Xiang Li, Danhao Ding, Ben Kao, Yizhou Sun, and Nikos Mamoulis. 2021. Leveraging meta-path contexts for classification in heterogeneous information networks. In *ICDE*. IEEE, 912–923.
[25] Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL/IJCNLP*, 4582–4597.
[26] Jianghao Lin, Weiwen Liu, Xinyi Dai, Weinan Zhang, Shuai Li, Ruiming Tang, Xiuqiang He, Jianye Hao, and Yong Yu. 2021. A graph-enhanced click model for web search. In *SIGIR*. ACM, 1259–1268.
[27] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. arXiv:2107.13586. Retrieved from https://arxiv.org/abs/2107.13586
[28] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks. In *WWW*. ACM, 417–428.
[29] Yihong Ma, Ning Yan, Jiayu Li, Masood S. Mortazavi, and Nitesh V. Chawla. 2024. HetGPT: Harnessing the power of prompt tuning in pre-trained heterogeneous graph neural networks. In *WWW*. ACM, 1015–1023.
[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*, Vol. 139. PMLR, 8748–8763.
[31] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *EMNLP-IJCNLP*. Association for Computational Linguistics, 3980–3990.
[32] Ridho Reinanda, Edgar Meij, and Maarten de Rijke. 2020. Knowledge graphs: An information retrieval perspective. *Foundations and Trends® in Information Retrieval* 14, 4 (Oct. 2020), 289–444.
[33] Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Improved semantic-aware network embedding with fine-grained word alignment. In *EMNLP*. Association for Computational Linguistics, 1829–1838.
[34] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. GPPT: Graph pre-training and prompt tuning to generalize graph neural networks. In *KDD*. ACM, 1717–1727.
[35] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in one: Multi-task prompting for graph neural networks. In *KDD*. ACM, 2120–2131.
[36] Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. 2016. A general framework for content-enhanced network representation learning. arXiv:1610.02906. Retrieved from https://arxiv.org/abs/1610.02906
[37] Xiangguo Sun, Jiawen Zhang, Xixi Wu, Hong Cheng, Yun Xiong, and Jia Li. 2023. Graph prompt learning: A comprehensive survey and beyond. arXiv:2311.16534. Retrieved from https://arxiv.org/abs/2311.16534
[38] Yizhou Sun and Jiawei Han. 2012. Mining heterogeneous information networks: A structural analysis approach. *ACM SIGKDD Explorations Newsletter* 14, 2 (2012), 20–28.
[39] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. CANE: Context-aware network embedding for relation modeling. In *ACL*. Association for Computational Linguistics, 1722–1731.
[40] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *KDD*. ACM, 1225–1234.
[41] Qunzhong Wang, Xiangguo Sun, and Hong Cheng. 2025. Does graph prompt work? A data operation perspective with theoretical analysis. In *ICML*, Vol. 267. PMLR/OpenReview.net.

[42] Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. 2021. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *KDD*. ACM, 1726–1736.
[43] Zenan Xu, Qinliang Su, Xiaojun Quan, and Weijia Zhang. 2019. A deep neural information fusion architecture for textual network embeddings. In *EMNLP-IJCNLP*. Association for Computational Linguistics, 4697–4705.
[44] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network representation learning with rich text information. In *IJCAI*. AAAI Press, 2111–2117.
[45] Xiaocheng Yang, Mingyu Yan, Shirui Pan, Xiaochun Ye, and Dongrui Fan. 2023. Simple and efficient heterogeneous graph neural network. In *AAAI*. AAAI Press, 10816–10824.
[46] Yaming Yang, Ziyu Guan, Jianxin Li, Wei Zhao, Jiangtao Cui, and Quan Wang. 2023. Interpretable and efficient heterogeneous graph convolutional network. *IEEE Transactions on Knowledge and Data Engineering* 35, 2 (2023), 1637–1650.
[47] Yaming Yang, Ziyu Guan, Zhe Wang, Wei Zhao, Cai Xu, Weigang Lu, and Jianbin Huang. 2022. Self-supervised heterogeneous graph pre-training based on structural clustering. In *NeurIPS*.
[48] Michihiro Yasunaga, Antoine Bosselut, Hongyu Ren, Xikun Zhang, Christopher D. Manning, Percy Liang, and Jure Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. In *NeurIPS*.
[49] Weiwei Yuan, Kangya He, Donghai Guan, Li Zhou, and Chenliang Li. 2019. Graph kernel based link prediction for signed social networks. *Information Fusion* 46 (2019), 1–10.
[50] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous graph neural network. In *KDD*. ACM, 793–803.
[51] Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *NAACL-HLT*. Association for Computational Linguistics, 5017–5033.
[52] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2021. Learning to prompt for vision-language models. arXiv:2109.01134. Retrieved from https://arxiv.org/abs/2109.01134
[53] Chenyi Zi, Haihong Zhao, Xiangguo Sun, Yiqing Lin, Hong Cheng, and Jia Li. 2024. ProG: A graph prompt learning benchmark. In *NeurIPS*.

Received 20 October 2025; revised 30 April 2026; accepted 26 May 2026