Pre-training on Large-Scale Heterogeneous Graph

Xunqiang Jiang[1], Tianrui Jia[1], Yuan Fang[2], Chuan Shi[1,3*], Zhe Lin[3], Hui Wang[3]
[1]Beijing University of Posts and Telecommunications
[2]Singapore Management University
[3]Peng Cheng Laboratory, Shenzhen, China
{skd621, jiatianrui}@bupt.edu.cn, yfang@smu.edu.sg,
shichuan@bupt.edu.cn, {linzh01,wangh06}@pcl.ac.cn

ABSTRACT

Graph neural networks (GNNs) emerge as the state-of-the-art representation learning methods on graphs and often rely on a large amount of labeled data to achieve satisfactory performance. Recently, in order to relieve the label scarcity issues, some works propose to pre-train GNNs in a self-supervised manner by distilling transferable knowledge from the unlabeled graph structures. Unfortunately, these pre-training frameworks mainly target at homogeneous graphs, while real interaction systems usually constitute large-scale heterogeneous graphs, containing different types of nodes and edges, which leads to new challenges on structure heterogeneity and scalability for graph pre-training. In this paper, we first study the problem of pre-training on a large-scale heterogeneous graph and propose a novel pre-training GNN framework, named PT-HGNN. The proposed PT-HGNN designs both the node- and schema-level pre-training tasks to contrastively preserve heterogeneous semantic and structural properties as a form of transferable knowledge for various downstream tasks. In addition, a relation-based personalized PageRank is proposed to sparsify a large-scale heterogeneous graph for efficient pre-training. Extensive experiments on one of the largest public heterogeneous graphs demonstrate that our PT-HGNN significantly outperforms various state-of-the-art baselines.

CCS CONCEPTS

• **Information systems → Data mining**.

KEYWORDS

Heterogeneous graph, self-supervised learning, pre-training

ACM Reference Format:
Xunqiang Jiang[1], Tianrui Jia[1], Yuan Fang[2], Chuan Shi[1,3][1], Zhe Lin[3], Hui Wang[3] . 2021. Pre-training on Large-Scale Heterogeneous Graph. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467396

*Corresponding author.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467396

1 INTRODUCTION

In recent years, as an emerging tool for learning on graph-structured data [33], graph neural networks (GNNs) learn powerful graph representations by recursively aggregating messages (*i.e.,* features) from neighboring nodes. They have been demonstrated to benefit a wide variety of graph mining tasks from node classification and link prediction [9, 15] to graph generation and graph classification [18]. However, the training of GNNs generally requires abundant task-specific labeled data in order to achieve competitive performance for each downstream task. On one hand, unfortunately, labeled data for many tasks are usually expensive or infeasible to obtain. On the other hand, a large amount of unlabeled graph structures are often readily available in various domains amid the digitization trend.

To alleviate the reliance on task-specific labeled data, inspired by pre-training techniques from computer version [4, 10] and natural language processing [6, 16], recent works propose to pre-train GNNs in a self-supervised manner by distilling transferable knowledge from the unlabeled graph structures [11, 12, 25]. In particular, the transferable knowledge aims to capture the inherent structural properties of the unlabeled graphs, which can be flexibly applied across different tasks to complement task-specific labeled data. The key differences between existing strategies of GNN pre-training boil down to the design of the self-supervised pre-training tasks and corpus. For example, Hu *et al.* [11] present various strategies which utilize node- and graph-level self-supervision on multiple graphs for pre-training the GNNs, while GPT-GNN [12] introduces a self-supervised attributed graph generation task on one large graph. Although the existing pre-training methods for GNNs achieve promising results, they are mainly designed for homogeneous graphs, lacking the capacity to capture the rich, heterogeneous semantic and structural properties of a heterogeneous graph during pre-training.

Heterogeneous graphs [26] have been commonly utilized for modeling complex systems, in which objects of different types interact with each other via various relations. For example, in an academic graph, authors can publish papers that appear in various conferences, and co-author with others; in an e-commerce graph, users can click or buy products in various shops, and shops can promote products. To handle the heterogeneous objects and interactions, various complex semantic patterns (*e.g.,* meta-path [27] and meta-graph [7]) have been proposed to capture the heterogeneous semantic and structural properties that are rich, inherent and vital on heterogeneous graphs. Thus, recent GNNs (without pre-training) start to utilize such semantic patterns to model heterogeneous graphs and achieve promising performance, implying that self-supervised pre-training tasks should also be designed to preserve the heterogeneous properties on a heterogeneous graph

as part of the transferable knowledge. However, existing GNN pre-training approaches have not attempted to encode such heterogeneity. Furthermore, real-world heterogeneous graphs are often large with million- or billion-scale nodes or edges. Hence, it becomes critical and timely to study the pre-training of GNNs on a large-scale heterogeneous graph.

Challenges and present work. In this paper, we take the first attempt to pre-train GNNs on a large heterogeneous graph, utilizing the rich semantic and structural properties as self-supervised information. However, the problem is non-trivial, presenting us with two key challenges, as follows.

(1) *How to capture the semantic and structural properties on a heterogeneous graph during pre-training?* Existing pre-training strategies [11, 12, 25, 35] are devised only for homogeneous graphs, treating all nodes and edges uniformly. However, a heterogeneous graph entails a variety of rich semantic and structural properties, which defines the varying characteristics of different types of objects. Taking an academic graph as example, a paper is characterized by not only its keywords but also its authors and venue, whereas an author is characterized by his/her institutes and papers. Moreover, different types of objects often manifest heterogeneous structural properties, *e.g.,* conference nodes generally have much higher degrees than author nodes. Thus, it is important to encapsulate such diverse semantic and structural characteristics in self-supervised pre-training tasks, in order to learn more precise and expressive transferable knowledge that can be tailored to different downstream tasks on a heterogeneous graph.

(2) *How to efficiently pre-train GNNs on a large-scale heterogeneous graph?* Real-word heterogeneous graphs can be enormous with billions of nodes and edges [36]. To ensure scalability to large graphs, existing work often considers two lines of approach: sampling [9, 13] and sparsification [3]. Sampling of neighboring nodes is often conducted online during training, whereas sparsification can be regarded as a form of graph compression to retain the most useful edges in an offline step. On a very large graph, online sampling still incurs a significant overhead in computing the distribution of nodes [3]. Thus, it is less desirable than sparsification, which does not introduce any overhead during training. In this paper, we resort to sparsification. However, existing sparsification methods, such as personalized PageRank and other centrality analysis, ignore the differences between various node and edge types, which could cause unwanted biases toward certain type of nodes—*e.g.,* the type of nodes with higher degrees will be indiscriminately deemed more important. Thus, it is vital to propose an edge sparsification process for large-scale heterogeneous graphs for efficient pre-training.

To tackle the above challenges, we present a novel framework of **P**re-**T**raining G**NN**s on **H**eterogeneous **G**raph, named **PT-HGNN**. It aims to preserve the inherent semantic and structural properties efficiently on a large-scale heterogeneous graph. For the first challenge, inspired by contrastive learning [34], we design a contrastive pre-training strategy to model the heterogeneity w.r.t. both semantics and structures. To be more specific, we introduce two contrastive strategies at the node and schema levels. At the node level, to enhance the subtle semantic difference between nodes, we generate relation-wise negative node samples—two nodes can form a negative sample if they satisfy the types involved in a specific relation and do not constitute a positive example under this relation, *e.g.,* author a_1 and paper p_2 with the "write" relation in Figure 1(a). At the schema-level, we generate negative subgraph samples guided by network schema, which effectively preserves high-order structures on heterogeneous graphs. Although complex semantic patterns (*i.e.,* meta-graph) are often used to capture higher-order graph heterogeneity, computation of their instances can be time-consuming. In contrast, as a unique high-order structure that defines a heterogeneous graph, network schema [27] is convenient to sample its instances, and schema instances naturally contain all types of nodes and relations in the graph. To address the second challenge, inspired by the previous graph sparsification studies [17, 29], we propose a relation-based personalized PageRank (PPR) to sparsify a large heterogeneous graph, as the original personalized PageRank [14] does not deal with graph heterogeneity. Particularly, in our relation-based sparsification, we distinguish various relations in the computation of personalized PageRank, thereby alleviating the bias introduced by certain types of nodes (*e.g.,* those with high degrees). The less biased sparsification can accelerate the pre-training procedure while preserving the heterogeneous nature of the graph.

Contributions. To summarize, we make the following major contributions in this work.

- This is the first attempt to pre-train GNNs on a large-scale heterogeneous graph, which is an important and practical problem with numerous application scenarios.
- We propose a novel pre-training strategy for GNNs, named PT-HGNN, which leverages both the node- and schema-level pre-training tasks to contrastively preserve heterogeneous semantic and structural properties as a form of transferable knowledge for various downstream tasks. At the same time, we propose a relation-based sparsification for efficient pre-training on a very large heterogeneous graph.
- We conduct extensive experiments on one of the largest public heterogeneous graphs, namely the Open Academic Graph (OAG) with more than 178 million nodes, and demonstrate that our PT-HGNN significantly outperforms various state-of-the-art baselines. Moreover, experiments validate that PT-HGNN is able to effectively transfer structural knowledge between different networks with similar structural properties.

2 RELATED WORK

Graph neural networks have received significant research interests due to the prevalence of graph-structure data [33]. They utilize neural networks to learn node representations on graphs, and belong to two major categories: spectral methods [2, 5] and message passing architectures to aggregate neighbors' features [9, 15]. Besides, some studies have attempted to deploy the GNNs on heterogeneous graphs [13, 32]. Wang *et al.* [32] have extended the attention mechanism for different meta-paths. Recently, Hu *et al.* proposes a heterogeneous graph transformer [13] that leverages multi-head attentions for different relation types to achieve better performance.

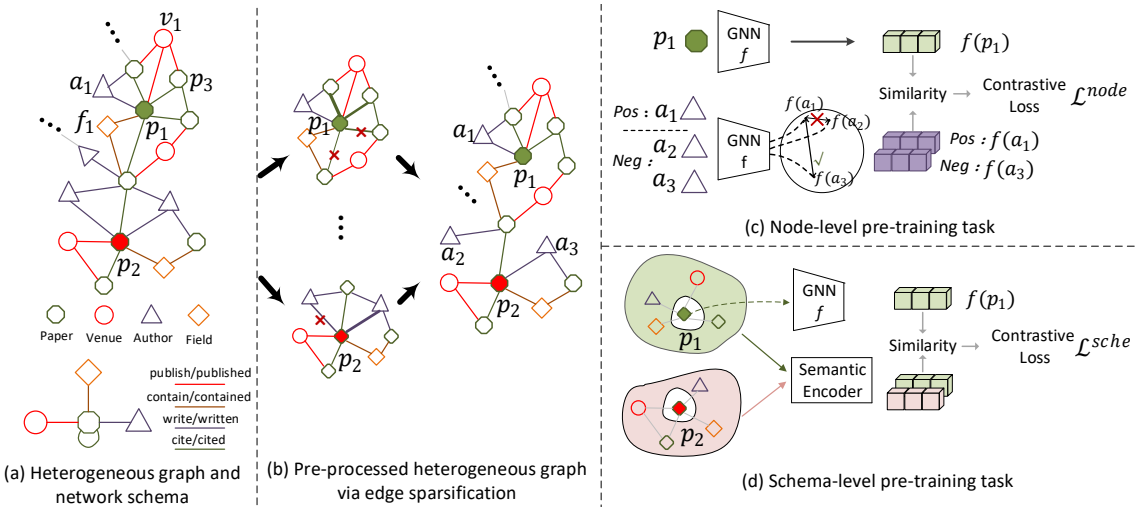


Figure 1: The overall framework of PT-HGNN.

To enable more effective learning, researchers have explored how to utilize the abundant unlabeled data for pre-training a GNN. Inspired by pre-training techniques in natural language processing [6] and computer vision [4, 10], recent studies [11, 12, 22, 25, 35] have been proposed to pre-train GNNs with self-supervised information. Boardly, these work are classified into two categories: contrastive method and generative method. For instances, Qiu *et al.* [25] and You *et al.* [35] propose various graph data augmentations to construct positive/negative samples for conducting contrastive learning, while GPT-GNN [12] introduces a self-supervised attributed graph generation task to pre-train a GNN. However, these methods usually focus on homogeneous graphs, which can not be directly applied to heterogeneous graphs.

3 PROPOSED PT-HGNN FRAMEWORK

In this section, we introduce our pre-training framework PT-HGNN on a large-scale heterogeneous graph. More specifically, we first elaborate on the design of the pre-training tasks for a heterogeneous graph. To preserve the heterogeneity, we propose both node- and schema-level pre-training tasks to respectively utilize node relations and the network schema, which encourages the GNN to capture heterogeneous semantic and structural properties. Second, we present our edge sparsification strategy on a large-scale heterogeneous graph for pre-processing. To avoid unwanted biases toward certain types of node, we propose a relation-based personalized PageRank to retain the most useful graph structures, in order to accelerate the pre-training procedure. Figure 1 shows the overall framework of the proposed PT-HGNN.

3.1 Pre-training Tasks on Heterogeneous Graph

A heterogeneous graph [26], denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \varphi\}$, is a form of graph, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges, respectively. It is also associated with a node-type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and an edge-type mapping function $\varphi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ denote the sets of node and edge types such that $|\mathcal{A}| + |\mathcal{R}| > 2$. Moreover, the network schema $T_{\mathcal{G}} = (\mathcal{A}, \mathcal{R})$ of a heterogeneous graph specifies the type constraints on the nodes and their relations, which can guide the exploration of heterogeneous structural contexts on the graph. Figure 1(a) shows an example of heterogeneous graph and its network schema with four types of node and edge.

For the design of pre-training tasks, our goal is to encode the inherent heterogeneity of nodes and edges in the transferable knowledge. In contrast to previous pre-training strategies [11, 12, 22, 25, 35], we need to consider the subtle semantic differences and complex heterogeneous structures. Thus, we employ both node instances and network schema instances as self-supervision, to differentiate the rich semantic relations and high-order heterogeneous structural contexts, respectively. Inspired by contrastive learning [25], we generate the positive and negative samples from two levels, to learn transferable knowledge for downstream tasks.

3.1.1 Node-level Pre-training Task. In existing methods, positive and negative samples are differentiated only by network structures, such as through a perturbation of the original graph structures and node features. However, a heterogeneous graph embodies rich semantics manifested in terms of multiple types of node and their relation (*i.e.,* edge), and thus it is crucial for the transferable knowledge to encode such semantics. Here, we design a node-level pre-training task to encode the semantics, which allows us to model pairwise relations between different types of node.

A relation between two nodes conveys important semantic information about them. On one hand, a positive triple $\langle u, R, v \rangle$ in a heterogeneous graph $\mathcal{G}$ means that nodes $u \in \mathcal{V}$ and $v \in \mathcal{V}$ are linked via a specific relation $R \in \mathcal{R}$ on $\mathcal{G}$. On the other hand, negative samples are obtained by replacing the nodes in a corresponding positive sample, as follows.

Negative Samples Selection. In previous contrastive learning [25, 35], the negative samples are only selected based on homogeneous graph structures (*i.e.,* two unlinked nodes), ignoring two important aspects. First, they do not constrain node types for a specific relation. For instance, for the "write" relation, the negative samples

should only be sampled from author-paper node pairs. Second, they do not consider the similarity of nodes themselves, in contradiction to the alignment principle [31]. Specifically, two nodes that are too similar should not simultaneously appear in the corresponding positive and negative samples under the same relation. For instance, consider a positive sample $\langle a_1, \text{"write"}, p_1 \rangle$ shown in Figure 1(c). a_2, though not linked to p_2, should not form a negative example with p_2 under the "write" relation, as a_2 and a_1 are too similar in their representations to serve as a good contrast. Instead, a_3, which is not linked to p_2 and is dissimilar to a_1, can form a more reasonable negative sample $\langle a_3, \text{"write"}, p_1 \rangle$. Accounting for both of the above aspects, we construct the negative samples in a relation-specific manner and consistent with the alignment principle. In specific, for a given positive triplet $\langle u, R, v \rangle$, we define the negative samples for a relation R as:

$$\mathcal{N}^{node}_{\langle u,R,v \rangle} = \{\langle u, R, v^- \rangle \mid \phi(v) = \phi(v^-), (u, v^-) \notin \mathcal{E}, Sim(v, v^-) \leq \delta\}, \tag{1}$$

where Sim is a function to measure the similarity between the node representations, and δ is a threshold for filtering out too similar nodes in violation of the alignment principle. Note that, to avoid getting only very easy negative samples, δ is set to a relatively large number.

Node-level Loss. Given a positive triple $\langle u, R, v \rangle$ and its corresponding negative samples that replace node v, we optimize the following InfoNCE loss [28] for node u:

$$\mathcal{L}^{node}_{u,R} = -\log \frac{\exp\left(\mathbf{h}_u^\top \mathbf{W}_R \mathbf{h}_v / \tau\right)}{\sum_{i \in \{v\} \cup \{w \mid \langle u,R,w \rangle \in \mathcal{N}^{node}_{\langle u,R,v \rangle}\}} \exp\left(\mathbf{h}_u^\top \mathbf{W}_R \mathbf{h}_i / \tau\right)}, \tag{2}$$

where $\mathbf{W}_R \in \mathbb{R}^{d \times d}$ is a learnable relation matrix for relation R and τ is a temperature hyper-parameter. Here, $\mathbf{h}_u$ indicates the representation vector of node u, which can be generated by any existing GNN architecture.

The above node-level pre-training tasks only capture the first-order semantics involving direct relations between nodes. To capture high-order semantic and structural properties, we resort to the schema-level pre-training, as follows.

3.1.2 Schema-level Pre-training Task. In order to incorporate the high-order heterogeneous semantic and structural contexts in a heterogeneous graph, a natural idea is to utilize high-order semantic patterns such as meta-structures [7, 27] and motifs. However, there are three major weakness of these semantic patterns for pre-training GNNs on a heterogeneous graph. (1) Meta-paths are relatively limited in the ability to express more complex high-order structures, due to its path-only structure, whereas motifs only capture high-order structures but limited in semantics. (2) For meta-graph or motif, it is intractable to find their instances especially when the meta-graph or motif is large. (3) The choice of meta-path and meta-graph relies on domain knowledge. However, for a heterogeneous graph, its network schema [27] is a unique defining structure that captures both high-order semantic and structural properties, and does not require any domain knowledge. Moreover, as network schema is essentially a template for the heterogeneous graph, its instances can be sampled easily.

Therefore, we utilize the network schema to strike a balance between capturing the heterogeneity and achieving efficiency on a large-scale heterogeneous graph. We generate schema instances to construct the positive and negative samples, to complement the first-order node-level samples.

Positive Network Schema Samples. Given a network schema $T_G = (\mathcal{A}, \mathcal{R})$, we can generate its instances. However, if we directly generate instances randomly following the network schema, the instances will be extremely imbalanced *w.r.t.* different node types, due to the fact that the degree of nodes for each type can vary dramatically (*e.g.,* the degree of a conference is much larger than a paper). To address the imbalance problem, we control the number of sampled schema instances of each type of node.

We consider the association between a node and a network schema instance. Formally, given a schema instance containing node u, let u be the target node and the other nodes in the instance be the context nodes of u. For example, for a schema instance $\mathbf{s} = \{p_1, a_1, f_1, v_1, p_3\}$ as shown in Figure 1(a), we say that node p_1 is a target node with context nodes $\{a_1, f_1, v_1, p_3\}$, or a_1 is a target node with context nodes $\{p_1, f_1, v_1, p_3\}$, and so on. Based on this definition, we consider the context nodes of target node u as the schema-level positive samples for u, denoted $\mathcal{P}^{sche}_u$, given by

$$\mathcal{P}^{sche}_u = \bigcup_{\mathbf{s} \in I(u)} \mathbf{s} \backslash \{u\}, \tag{3}$$

where $I(u)$ denotes the set of all schema instances containing node u. Intuitively, the context nodes capture not only the structural contexts of the target node, but also semantic properties of different types of node and edge.

Negative Network Schema Samples. To construct the schema-level negative samples for target node u, we follow two approaches: 1) if two network schema instances are generated from two different target nodes of the same type, we treat them as negative schema samples of each other; and 2) we design a dynamic queue [10] for storing the negative network schema samples.

In the first approach, for the nodes in a batch denoted as $\mathcal{V}_B$, we obtain the negative schema instances $\mathcal{N}^1_u$ for target node u as

$$\mathcal{N}^1_u = \{\mathcal{P}^{sche}_{u^-} \mid u^- \in \mathcal{V}_B, u \neq u^-, \phi(u) = \phi(u^-)\}. \tag{4}$$

In the second approach, for the sake of getting more negative schema samples, a direct idea is to randomly select the nodes for each type to construct the network schema instances. However, negative instances generated from this method will be easily distinguishable from the positive schema samples, in which nodes in such negative samples are more likely to be unrelated. Thus, to avoid randomly generated negative samples, we resort to actual schema instances from the previous batch by designing a dynamic queue to store the network schema instances for more representative instances. Then, we have the negative schema instances $\mathcal{N}^2_u$ from the dynamic queue as:

$$\mathcal{N}^2_u = \{\mathcal{P}^{sche}_v \mid \phi(u) = \phi(v), v \in \mathcal{V}^{t-1}_B\}, \tag{5}$$

where $\mathcal{V}^{t-1}_B$ is the node set of the previous batch. It is worth noticing that the queue is initialized as an empty set in the beginning and updated during the training procedure. Therefore, we obtain the overall schema-level negative samples as follows:

$$\mathcal{N}^{sche}_u = \mathcal{N}^1_u \cup \mathcal{N}^2_u. \tag{6}$$

Encoding Context Nodes and Schema-Level Loss. Directly predicting the proximity between target node u and its context nodes $\{v_1, v_2, \cdots\}$ cannot capture the heterogeneity of nodes. To account for nodes of different types, we devise one encoder for each type of nodes to learn node representations. Specifically, we learn the node embedding of target node v_i with an encoder $\text{Enc}^{\phi(v_i)}$ for node type $\phi(v_i)$:

$$\mathbf{e}_{v_i} = \text{Enc}^{\phi(v_i)}(\mathbf{h}_{v_i}), \tag{7}$$

where each encoder $\text{Enc}(\cdot)$ represents an MLP. Then, for the target node u, we generate its context embedding via a pooling function over the context nodes $\{v_1, v_2, \cdots\}$, denoted as $\mathbf{c}_u^s$:

$$\mathbf{c}_u^s = \text{Pool}(\mathbf{e}_{v_1}, \mathbf{e}_{v_2}, \cdots), \tag{8}$$

where $\text{Pool}(\cdot)$ averages the embeddings of context nodes.

Given the schema-level samples, we optimize the likelihood that the target node u associates with context nodes in the positive samples and does not related to those in the negative samples:

$$\mathcal{L}_u^{sche} = \sum_{\mathbf{s}^+ \in \mathcal{P}_u^{sche}} \log \frac{\exp\left(\mathbf{h}_u^\top \mathbf{c}^{\mathbf{s}^+}/\tau\right)}{\sum_{\mathbf{s} \in \{\mathbf{s}^+\} \cup \mathcal{N}_u^{sche}} \exp\left(\mathbf{h}_u^\top \mathbf{c}^{\mathbf{s}}/\tau\right)}, \tag{9}$$

where τ is a temperature hyper-parameter.

3.2 Sparsification of Large-Scale Heterogeneous Graph

To pre-train a better GNN, the key is to leverage a large-scale heterogeneous graph. Existing scalable GNNs utilize sampling and sparsification in the online and offline steps, respectively. However, on large-scale graphs, online sampling still incurs a significant overhead in computing the distribution of nodes [3]. Thus, we resort to offline sparsification to retain the most important edges in the graph. Inspired by previous studies [1, 17, 29], personalized PageRank can be utilized to preserve more effective neighborhood. However, due to the heterogeneity of graph, existing personalized PageRank on homogeneous graphs is not applicable. Thus, we propose a relation-based personalized PageRank for edge sparsification.

3.2.1 Personalized PageRank on Heterogeneous Graph. As we mentioned above, personalized PageRank (PPR) is a good tool to obtain more meaningful neighbors for a node. To evaluate the personalized PageRank for a certain node, it relies on the node degree and adjacency matrix to calculate the transition probability. However, on a heterogeneous graph, the transition probability can be highly skewed toward certain types of node. For example, the transition probability from conference node (high out-degree) and paper node (low out-degree), will be extremely different due to their distinct structural role in an academic graph, which leads to unwanted biases in the PageRank scores.

To alleviate the biases caused by the heterogeneous structures, we design a power-iteration method to compute the personalized PageRank score for a relation R. For the sake of simplicity, we take $A_1 \xrightarrow{R} A_2$ for illustration, which defines a relation R between nodes with types A_1 and A_2. We obtain personalized PageRank Π^R for a relation R by iteratively updating the following:

$$\begin{aligned} \Pi^R &= \alpha \mathbf{I} + (1-\alpha) S \Pi^{R^{-1}} \\ \Pi^{R^{-1}} &= \beta \mathbf{I} + (1-\beta) S^T \Pi^R \end{aligned} \tag{10}$$

where $S = D_{A_1}^{-\frac{1}{2}} A^R D_{A_2}^{-\frac{1}{2}}$, D_{A_i} denotes a diagonal matrix whose diagonal contains the degrees of all nodes of A_i type, A^R represents the adjacency matrix for nodes with type A_1 and A_2 under the relation R, Π^R denotes a matrix for pairwise personalized PageRank scores under the relation R, R^{-1} is the inverse relation of R, and α, β are hyper-parameters for controlling the convergence. Thus, the relation-based personalized PageRank can handle different node and edge types without skewing to a particular type.

However, the computation of Eq. 10 depends on matrix multiplication, which is infeasible for a large-scale graph. Inspired by previous studies [1], we obtain the personalized PageRank for a relation R using a equivalent random walk formulation. Further details are provided in Appendix A.

3.2.2 Edge Sparsification. Through the above process, we can obtain a dense matrix Π^R with pairwise personalized PageRank scores for each relation R, in contrast to the sparse adjacency matrix A^R under the same relation. Note that the values in Π^R represent the pairwise influence between all pairs of nodes in relation R, which typically are highly localized [23]. Spatial localization allows us to simply truncate small values of Π^R and recover sparsity. Similar to previous studies [1], we use the top-k entries with the highest mass per column and set all other entries to zero in Π^R. Subsequently, as shown in Figure 1(b), for each relation R, we define a new adjacency matrix $\tilde{A}^R$ as follows:

$$\tilde{A}_{uj}^R = \begin{cases} 1, & \text{if } \Pi_{uj}^R > 0 \text{ and } (u,j) \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases}. \tag{11}$$

The edge sparsification for the heterogeneous graph $\mathcal{G}$ will be done as a pre-processing step. Subsequently, we obtain the sparsified graph $\mathcal{G}'$ for pre-training, which can accelerate the aggregation operation of GNNs.

3.3 Pre-training Pipeline

Figure 1 shows the overall pipeline of the pre-training framework (PT-HGNN). Given a large-scale heterogeneous graph $\mathcal{G}$, PT-HGNN first conducts the edge sparsification via the relation-based personalized PageRank to obtain a more representative and sparsified graph $\mathcal{G}'$, as show in Figure 1(b). Then, PT-HGNN employs the pre-training tasks at both the node and relation levels on the sparsified $\mathcal{G}'$, in which the two pre-training tasks are jointly optimized to capture the heterogeneous semantic and structural properties as the transferable knowledge. In other words, we minimize the following loss on $\mathcal{G}'$:

$$\mathcal{L} = \mathcal{L}^{node} + \lambda \mathcal{L}^{sche}, \tag{12}$$

where λ is a balancing coefficient. At last, we optimize the model via the AdamW optimizer [21] with Cosine Annealing Learning Rate Scheduler [20]. The detailed algorithm and time complexity of the pre-training pipeline are provided in Appendix B.

Table 1: Statistics of Open Academic Graph Dataset

Dataset	#nodes	#edges	#venues	#papers	#fields	#authors	#institutes	#P-V	#P-P	#P-F	#P-A	#A-I
CS	11,918,983	107,263,811	27,433	5,597,605	289,930	5,985,759	18,256	5,597,606	31,441,552	47,462,559	15,571,614	7,190,480
Mater	4,552,941	42,161,581	15,141	2,442,235	79,305	2,005,362	10,898	2,442,235	13,011,272	19,119,947	5,582,765	2,005,362
Engin	5,191,920	36,146,719	19,867	3,239,504	99,444	1,819,100	14,005	3,239,504	4,848,158	22,498,822	3,741,135	1,819,100
Chem	12,158,967	159,537,437	19,142	7,193,321	183,782	4,748,812	13,910	7,193,321	74,018,600	57,162,528	16,414,176	4,748,812
OAG	178,663,987	2,236,196,802	53,073	89,606,257	615,288	88,364,081	25,288	89,606,258	1,021,237,518	657,049,405	300,853,688	167,449,933

4 EXPERIMENTS

In this section, we first conduct experiments on large heterogeneous graphs to evaluate model performance, and then investigate the knowledge transferability of PT-HGNN on different graphs and the underlying mechanism with two ablated models. Lastly, we explore the impact of different model settings on task performance.

4.1 Experiment Settings

Datasets and Tasks. We conduct experiments on Open Academic Graph (OAG) [36], which is consitituted by papers (P), authors (A), venues (V), institutes (I), fields (F) and their relations with 178 million nodes and 2.236 billion edges. As far as we know, this is the largest publicly available heterogeneous graph. To test the generalization ability and transferability of the proposed pre-training framework, we also construct four representative domain-specific subgraphs from OAG: Computer Science (CS), Material Science (Mater), Engineering (Engin) and Chemistry (Chem). These graphs have diverse statistics listed in Table 1, which also exhibit large differences in graph properties as shown in Appendix C. We consider the prediction of Paper–Field, Paper–Venue, and Author Name Disambiguation (Author ND) as three downstream tasks used in prior works [12, 13]. The model performance is evaluated by NDCG and MRR, which are widely adopted performance metrics [19].

Pre-Training and Fine-Tuning Setting. We pre-train a GNN model and use its output node embeddings as input features for the downstream tasks. Then we fine-tune the GNN models according to the specific downstream tasks on an unseen fine-tuning dataset and evaluate the model performance. In OAG dataset, the publication data of papers ranges from 1900 to 2019. Accordingly, we use the data with publication information before 2014 for pre-training. In contrast, the data since 2014 are used for fine-tuning on different downstream tasks. To be more specific, in the fine-tuning stage, we use 10% data from 2014 to 2016 for training and 10% data in 2017 for validation. The data after 2018 are used for testing.

Baselines. We compare our proposed PT-HGNN with a series of state-of-the-art baselines, as follows.

- No pre-train method trains a GNN model for the downstream tasks on the fine-tuning graph directly.
- EdgePred [9] predicts whether there exists a link between two nodes, which is an unsupervised method that forces linked nodes to have similar node embedding.
- DGI [30] maximizes local mutual information across the graph's patch representations.
- ContextPred [11] maps nodes appearing in similar structural contexts to nearby embeddings.
- GraphCL [35] proposes four graph data augmentation method to conduct contrastive learning, in which node dropping and subgraphs are employed as the pre-training strategies in the experiment.
- GPT-GNN [12] is generative pre-training model for GNNs, which reconstructs the attributes and the structure of the input graph to learn the transferable knowledge from the input graph.

Implementation details. We employ the state-of-the-art heterogeneous GNN method HGT [13] as the base model for our method PT-HGNN and other baselines. We implement the base model with PyTorch Geometric (PyG) package [8]. We set the hidden dimension as 400, the number of heads as 8, and the number of layers as 3. Besides, we optimize the model via AdamW optimizer [21] with Cosine Annealing Learning Rate Scheduler [20] with 200 epochs and choose the model parameters with the lowest validation loss as the pre-trained model in the pre-training procedure. For fair comparison, the above parameters follow the setting of previous study [12]. For the implementation of our pre-training framework PT-HGNN, we set δ in Eq.(1) as 0.99. The maximum number of negative samples in Eq. (1) and Eq. (4) are set to 256 and 512, respectively. In the pre-training procedure, we will remove some negative samples randomly if the number of negative samples exceeds the maximum setting. For the balancing coefficient λ in Eq. (12), we set it as 0.1 in the pre-training procedure. The temperature hyper-parameter τ in Eq.(2) and Eq.(9) are set to 0.2. In the fine-tuning stage, for fair comparison, we fine-tune the model using the same optimization setting on the downstream tasks for ten times and report the mean and standard deviation of the test performance. More experiment details can be found in Appendix D.

4.2 Performance Comparison on Downstream Tasks

In this experiment, we compare PT-HGNN to baseline methods in the downstream tasks on the whole OAG graph and the four domain-specific subgraphs, respectively. We evaluate the model performance by pre-training and fine-tuning on the same graphs but with different time periods as described in Section 4.1. Table 2 demonstrates the link prediction and node classification performance on the five datasets.

Overall, PT-HGNN achieves relative performance gains of 22.02% over the base model (*i.e.,* HGT) without pre-training on OAG. Moreover, our PT-HGNN consistently yields the best performance among all methods, leading to an average improvement of 4.98% compared to the second best baseline method. These improvements indicate that our proposed pre-training strategy is capable of exploiting transferable information and graph properties on heterogeneous graphs, which are beneficial to the downstream tasks. We also observe that GPT-GNN achieves the second best performance due to the fact that its edge generation pre-training task can also be

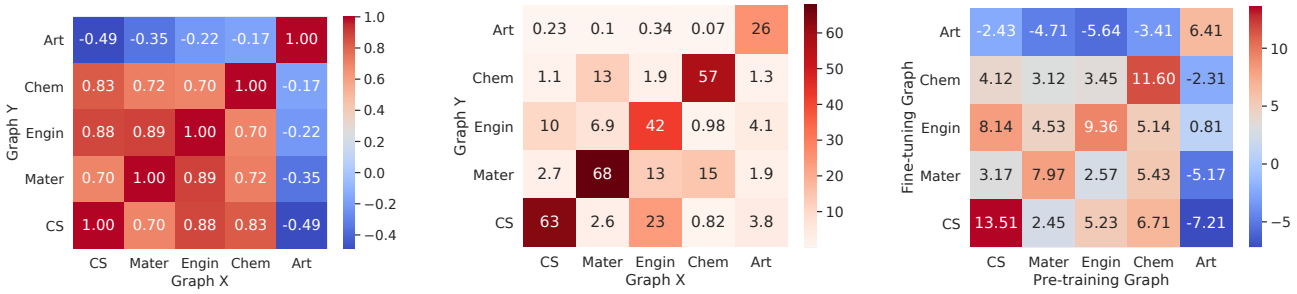
Table 2: Performance comparison between our method and baselines (best result in bold, and second best underlined).

Dataset	Downstream Task		No pre-train	EdgePred	DGI	ContextPred	GraphCL	GPT-GNN	PT-HGNN	Improv.
CS	Paper–Field	NDCG	27.42±0.42	31.37±0.32	32.82±0.67	33.15±0.71	32.64±0.65	35.24±0.47	**36.04**±0.37	2.27%
		MRR	23.17±0.45	32.13±0.52	33.43±0.81	33.24±0.57	33.24±0.67	33.57±0.71	**37.76**±0.42	12.48%
	Paper–Venue	NDCG	27.76±0.56	35.77±0.59	34.23±0.71	34.30±0.92	32.11±0.69	36.15±0.53	**38.81**±0.51	7.35%
		MRR	11.39±0.37	16.34±0.47	16.21±0.62	17.66±0.81	16.29±0.49	19.13±0.65	**21.19**±0.45	10.76%
	Author ND	NDCG	76.27±0.53	79.41±0.68	81.38±0.93	79.22±0.72	79.95±0.89	80.20±0.51	**82.19**±0.60	2.48%
		MRR	54.82±0.49	59.06±0.74	58.98±0.79	60.23±0.83	60.55±0.74	60.94±0.52	**63.38**±0.38	4.00%
Mater	Paper–Field	NDCG	37.44±0.77	44.47±0.67	43.32±0.66	41.94±0.32	43.89±0.54	44.77±0.47	**46.61**±0.64	4.01%
		MRR	35.56±0.54	47.69±0.75	46.21±0.81	44.11±0.69	46.24±0.80	48.24±0.35	**51.11**±0.51	5.94%
	Paper–Venue	NDCG	37.94±0.56	43.73±0.59	43.32±0.68	43.26±0.49	42.76±0.44	44.23±0.63	**47.88**±0.52	8.25%
		MRR	18.78±0.49	23.23±0.33	26.77±0.42	27.32±0.44	24.68±0.34	28.39±0.52	**29.54**±0.41	4.05%
	Author ND	NDCG	71.45±0.84	71.52±0.82	70.49±0.71	71.44±0.89	71.52±0.66	**72.75**±0.58	71.99±0.72	-1.04%
		MRR	47.46±0.63	48.49±0.56	47.38±0.42	46.32±0.49	48.22±0.59	49.92±0.40	**51.57**±0.47	3.31%
Engin	Paper–Field	NDCG	28.44±0.32	34.33±0.46	34.40±0.56	34.88±0.60	35.20±0.60	35.94±0.44	**37.78**±0.53	5.12%
		MRR	23.90±0.42	36.86±0.45	36.43±0.55	37.42±0.46	35.74±0.49	38.28±0.39	**40.24**±0.44	5.12%
	Paper–Venue	NDCG	41.44±0.67	47.25±0.88	47.53±0.53	46.81±0.69	45.29±0.93	47.96±0.71	**50.27**±0.65	4.81%
		MRR	23.67±0.76	27.65±0.59	29.29±0.78	30.14±0.88	29.13±0.92	29.79±0.65	**31.12**±0.64	4.46%
	Author ND	NDCG	73.77±1.02	73.92±0.96	74.78±0.74	74.32±0.89	74.56±0.75	75.22±0.82	**76.71**±0.70	1.94%
		MRR	49.37±0.52	50.87±0.57	51.29±0.63	50.23±0.72	50.37±0.51	52.66±0.64	**54.47**±0.59	3.44%
Chem	Paper–Field	NDCG	31.74±0.46	36.53±0.49	38.45±0.66	39.27±0.49	39.90±0.42	41.76±0.39	**42.85**±0.50	2.61%
		MRR	23.90±0.37	44.66±0.39	46.25±0.47	47.02±0.42	45.39±0.58	46.70±0.49	**48.70**±0.62	4.11%
	Paper–Venue	NDCG	30.39±0.60	42.23±0.57	43.56±0.53	44.11±0.51	42.73±0.59	43.05±0.62	**46.81**±0.43	6.12%
		MRR	13.68±0.32	22.13±0.33	23.27±0.35	22.85±0.44	21.84±0.56	24.19±0.62	**27.64**±0.49	14.26%
	Author ND	NDCG	75.19±0.94	76.71±0.82	75.69±0.69	77.60±0.76	77.65±0.86	78.31±0.85	**80.09**±0.87	2.27%
		MRR	55.61±0.62	58.33±0.59	57.00±0.44	59.20±0.59	57.57±0.38	60.30±0.55	**62.91**±0.50	6.48%
OAG	Paper–Field	NDCG	32.33±0.36	38.03±0.33	37.12±0.42	38.40±0.49	39.32±0.30	40.76±0.40	**42.33**±0.62	3.85%
		MRR	28.15±0.48	44.23±0.56	42.96±0.43	43.15±0.55	45.65±0.60	45.70±0.41	**47.29**±0.49	3.48%
	Paper–Venue	NDCG	42.28±0.50	43.25±0.61	44.23±0.53	43.07±0.74	42.66±0.66	44.05±0.75	**47.13**±0.68	6.56%
		MRR	22.76±0.37	23.40±0.35	24.38±0.35	24.12±0.42	25.03±0.48	25.19±0.45	**26.75**±0.57	6.19%
	Author ND	NDCG	76.52±1.13	78.01±0.86	77.98±0.93	77.88±0.72	78.11±0.93	79.33±0.87	**79.99**±0.92	0.83%
		MRR	54.65±0.53	58.00±0.63	58.30±0.48	57.49±0.60	58.22±0.53	59.08±0.52	**61.32**±0.55	3.79%

adopted for heterogeneous graphs, while its performance is significantly worse than our PT-HGNN. This demonstrates the superiority of our proposed pre-training tasks in capturing the heterogeneity on graphs. In addition, among the baselines, different pre-training methods work well on different datasets, *e.g.,* EdgePred for Material Science and ContextPred for Chemistry. This difference implies that those baselines may only capture a portion of local information and structural contexts which make their performance vary dramatically in different datasets. In all, the proposed node- and schema- level pre-training tasks enable our PT-HGNN to make full use of heterogeneous semantic and structural properties on heterogeneous graphs, which accounts for the performance gain of PT-HGNN over the other baselines.

4.3 Knowledge Transfer on Different Graphs

We investigate how the network structures affect the ability of knowledge transfer from pre-training to fine-tuning, and examine the applicability of our proposed pre-training strategies. In order to exploit diverse network structures, in this experiment, we add the Art graph with unique network structures from the art field of OAG, whose details can be found in Appendix C. We compute the correlation between graphs using a series of commonly used graph property metrics [24] listed in Table 6 in Appendix C, so as to quantify the structural difference between graphs. In all, a higher correlation value between two graphs means the higher similarity of their network structures. Figure 2(a) shows a heatmap of the pairwise correlation among five representative graphs, from which we observe that the four graphs out of a total number of five, namely, CS, Chemistry, Engineering and Materials, have similar network structures, whereas the Art graph is significantly different in its network structures compared to the other graphs. This difference in network structures is also verified by the citation coefficient [36], which measures the percentage of publications in graph Y that have citations in graph X, as shown in Figure 2(b).



(a) Correlation based on graph properties (b) Citation coefficient (%) (c) MRR gain (%) over no pre-training

Figure 2: Visualization of pairwise correlation and knowledge transfer among five graphs.

Table 3: Analysis of different ablated models on various downstream tasks on the CS graph.

Downstream Task		No pre-train	PT-HGNN$_{node}$	PT-HGNN$_{sche}$	PT-HGNN
Paper–Field	NDCG	27.42	35.80	35.16	36.04
	MRR	23.17	36.82	36.21	37.76
Paper–Venue	NDCG	27.76	36.23	35.24	38.81
	MRR	11.39	20.42	18.92	21.19
Author ND	NDCG	76.27	80.41	81.25	82.19
	MRR	54.82	60.57	62.02	63.38

Next, we evaluate the model performance by pre-training with one graph and fine-tuning with another for the Author ND task. The MRR improvement of our proposed method over the one without pre-training is shown in Figure 2(c), which leads to the following findings. Knowledge transferring from pre-training to fine-tuning does not guarantee a gain in performance, and generally speaking, a positive correlation value between graphs results in positive transferring and vice versa. Moreover, a higher similarity of network structures between graphs tend to give rise to larger performance improvement. From the above observations, we can come to the conclusion that the proposed pre-training strategy is applicable if the two graphs for pre-training and fine-tuning, respectively, have similar network structures with a positive correlation value.

4.4 Ablation Study

We experimentally evaluate the effect of node- and schema-level pre-training tasks on heterogeneous graphs. Hence, we conduct an ablation study and consider two ablated variants of PT-HGNN named PT-HGNN$_{node}$ and PT-HGNN$_{sche}$. Here, PT-HGNN$_{node}$ only includes the node-level pre-training task, while PT-HGNN$_{sche}$ only incorporates the schema-level pre-training task. Specifically, we consider the prediction of Paper–Field, Paper–Venue, and Author ND as three downstream tasks on the CS graph.

In Table 3, the two ablated variants achieve significant improvement over the no pre-train model, while they still perform worse than the complete pre-training strategy, *i.e.,* PT-HGNN. These results illustrate the benefits of the node- and schema-level pre-training tasks. In particular, PT-HGNN$_{node}$ models the pairwise node interaction with semantic information, which obtains better performance than PT-HGNN$_{sche}$ in the link prediction experiments (*i.e.,* Paper-Field and Paper-Venue). Compared to PT-HGNN$_{node}$, the PT-HGNN$_{sche}$ significantly improves the node classification performance by focusing on modeling the structure context in a heterogeneous graph. Putting it all together, the combination of PT-HGNN$_{node}$ and PT-HGNN$_{sche}$ offers strong capability in learning both the link prediction and node classification tasks.

4.5 Model Analysis

Lastly, we investigate the performance of PT-HGNN with different fine-tuning strategies, as well as the impact of edge sparsification.

Freezing vs. Full Fine-tuning. The main goal of pre-training is to learn transferable weights. Thus, we evaluate two different training modes for downstream tasks, named freezing mode and full fine-tuning mode. In the freezing mode denoted by PT-HGNN(FE), we freeze the parameters of the pre-trained model during fine-tuning stage, and treat it as a feature extractor (FE) only. Then, we just train the downstream classifier with the output embeddings from the base GNN. In the full fine-tuning mode (i.e., our PT-HGNN), we train the base GNN with the downstream classifier in an end-to-end manner. We compare the performance of freezing mode, full fine-tuning mode and the no pre-train mode. As shown in Table 4, PT-HGNN(FE) achieves better performance than the no pre-train model, which demonstrates that the proposed pre-training strategies are able to capture the transferable knowledge. Moreover, the performance of PT-HGNN in freezing mode exhibits competitive performance to that of the full fine-tuning mode. In all, the experimental results indicate that our pre-training strategies can capture transferable knowledge.

Impact of edge sparsification. We conduct further experiments to examine the efficiency of the PPR-based edge sparsificaiton operation. Specifically, we evaluate the three downstream tasks on the CS graph with and without the sparsification method. Table 5 demonstrates that pre-training on the pre-processed heterogeneous graph $\mathcal{G}'$ achieve competitive performance to that on the original input heterogeneous graph $\mathcal{G}$. This is mainly because the PPR strategy in PT-HGNN can preserve more meaningful edges and reduce some noisy edges in the pre-processing stage. Moreover, with the edges sparsification based on personalized PageRank, the training efficiency is increased by 41.97% on average, showing that this strategy effectively expedites the pre-training process.

Table 4: Performance of PT-HGNN in freezing and full fine-tuning modes on the CS graph.

Downstream Task		No pre-train	PT-HGNN (FE)	PT-HGNN
Paper–Field	NDCG	27.42	32.81	36.04
	MRR	23.17	32.50	37.76
Paper–Venue	NDCG	27.76	35.93	38.81
	MRR	11.39	18.45	21.19
Author ND	NDCG	76.27	81.41	82.19
	MRR	54.82	62.15	63.38

Table 5: Performance and time efficiency of our model with PPR and without PPR-based sparsification on the CS graph.

Downstream Task		No PPR	PPR	Improv.
Paper–Field	NDCG	**36.54**	36.04	-1.38%
	MRR	**38.12**	37.76	-0.95%
Paper–Venue	NDCG	37.82	**38.81**	2.62%
	MRR	20.42	**21.19**	3.77%
Author ND	NDCG	80.87	**82.19**	1.63%
	MRR	60.09	**63.38**	5.48%
Efficiency	Time per batch (s)	64.2	**37.9**	41.97%

5 CONCLUSION

In this work, we take the first attempt to pre-train GNNs on a large-scale heterogeneous graph, and introduce a pre-training framework named PT-HGNN. First, to preserve the heterogeneity, we propose both node- and schema-level pre-training tasks to utilize node relations and the network schema, respectively, which enables the GNN to capture heterogeneous semantics and structural properties. Second, to pre-train on large-scale heterogeneous graphs, we present an edge sparsification strategy via relation-based personalized PageRank, which retains meaningful graph structures while accelerating the pre-training procedure. Extensive experiments on one of the largest heterogeneous graphs, OAG, demonstrate the superior ability of our PT-HGNN to transfer knowledge to various downstream tasks via pre-training.

6 ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 61772082, 61702296, 62002029), Key fields R&D project Of Guangdong Province (No. 2020B0101380001). All opinions, findings, conclusions and recommendations are those of the authors and do not reflect the views of the funding agencies.

REFERENCES

[1] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate pagerank. In *KDD*. 2464–2473.
[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.
[3] Julian Busch, Jiaxing Pi, and Thomas Seidl. 2020. PushNet: Efficient and Adaptive Neural Message Passing. In *ECAI*. 1039–1046.
[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*. PMLR, 1597–1607.
[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*. 3837–3845.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.
[7] Yuan Fang, Wenqing Lin, Vincent Wenchen Zheng, Min Wu, Kevin Chen-Chuan Chang, and Xiaoli Li. 2016. Semantic proximity search on graphs with metagraph-based learning. In *ICDE*. 277–288.
[8] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop*.
[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.
[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*. 9729–9738.
[11] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. 2020. Strategies for Pre-training Graph Neural Networks. In *ICLR*.
[12] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In *KDD*. 1857–1867.
[13] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *TheWebConf*. 2704–2710.
[14] Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *WWW*. 271–279.
[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[16] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.
[17] Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *KDD*. 631–636.
[18] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter W Battaglia. 2018. Learning Deep Generative Models of Graphs. In *ICLR*.
[19] Tie-Yan Liu. 2011. Learning to rank for information retrieval. (2011).
[20] Ilya Loshchilov and Frank Hutter. 2016. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).
[21] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).
[22] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to Pre-train Graph Neural Networks. In *AAAI*. 4276–4284.
[23] Huda Nassar, Kyle Kloster, and David F Gleich. 2015. Strong localization in personalized PageRank vectors. In *International Workshop on Algorithms and Models for the Web-Graph*. 190–202.
[24] Mark EJ Newman. 2003. The structure and function of complex networks. *SIAM Rev.* 45, 2 (2003), 167–256.
[25] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *KDD*. 1150–1160.
[26] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. 2017. A Survey of Heterogeneous Information Network Analysis. In *TKDE*. 17–37.
[27] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. In *VLDB*. 992–1003.
[28] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748* (2018).
[29] Andrea Vattani, Deepayan Chakrabarti, and Maxim Gurevich. 2011. Preserving Personalized Pagerank in Subgraphs.. In *ICML*. 793–800.
[30] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *ICLR*.
[31] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*. 9929–9939.
[32] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. 2022–2032.
[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2020. A Comprehensive Survey on Graph Neural Networks. *IEEE TNNLS* 32, 1 (2020), 4–24.
[34] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*. 3733–3742.
[35] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. In *NeurIPS*. 5812–5823.
[36] Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, et al. 2019. OAG: Toward linking large-scale heterogeneous entity graphs. In *KDD*. 2585–2595.

A RELATION-BASED PERSONALIZED PAGERANK WITH RANDOM WALK

The algorithm of relation-based personalized PageRank (PPR) with random walk is described in Algorithm 1. The core idea of this algorithm is to compute an approximate personalized PageRank vector per node per relation type using random walk, and then assemble the resulted vectors into personalized PageRank matrices according to the relation types. Note that in Algorithm 1, we distinguish the relations by both their types and directions, *e.g.,* a relation denoting that an author writes a paper is different from the relation showing that a paper is written by an author.

Algorithm 1 Relation-Based PPR with Random Walk

Require: Heterogeneous graph $\mathcal{G}$, node t, source type A_1, target type A_2, relation R, max residual ϵ, out-degree vector $\boldsymbol{d}$ and teleport prob α, β

Initialize the approximate pagerank vector $\boldsymbol{\pi}^R$, the residual vector for source type $\boldsymbol{r}^{A_1} = \alpha \cdot \mathbf{e}_t$ and target type $\boldsymbol{r}^{A_2} = \mathbf{0}$
while $\exists v$ *s.t.* $\frac{\boldsymbol{r}_v^{A_1}}{\boldsymbol{d}_v} \geq \alpha \cdot \epsilon$ or $\frac{\boldsymbol{r}_v^{A_2}}{\boldsymbol{d}_v} \geq \beta \cdot \epsilon$ **do**
  if $\phi(v) = A_2$ **then**
    $\boldsymbol{\pi}^R$ += $\boldsymbol{r}_v^{A_2}$
    for $u \in Nbr_{\mathcal{G}}^{R^{-1}}(v)$ **do**
      $r_u^{A_1}$ += $(1-\beta) \cdot r_v^{A_2}/\sqrt{d_u d_v}$
    end for
    $\boldsymbol{r}_v^{A_2} = 0$
  else
    for $u \in Nbr_{\mathcal{G}}^{R}(v)$ **do**
      $r_u^{A_2}$ += $(1-\alpha) \cdot r_v^{A_1}/\sqrt{d_u d_v}$
    end for
    $\boldsymbol{r}_v^{A_1} = 0$
  end if
end while
return $\boldsymbol{\pi}^R$ for node t in relation R

B PRE-TRAINING PIPELINE AND TIME COMPLEXITY

The training algorithm for PT-HGNN is outlined in Algorithm 2. We analyze the time complexity of our pre-training pipeline from the following two aspects.

Time complexity of conducting pre-training tasks. This time complexity is mainly induced by the message passing and aggregation mechanism of GNNs, node-level pre-training task and schema-level pre-training task. The time for the message passing and aggregation mechanism depends on the base GNN architecture. The time for the node-level pre-training task and the schema-level pre-training task is linearly related to the number of samples. To be more specific, the time complexity of pre-training is determined by the pre-defined number of negative samples collected for each positive sample. The overall time complexity for pre-training is described by $O\left(X + |\mathcal{V}|\left(|\overline{\mathcal{E}}|k_1 + \overline{|\mathcal{P}^{sche}|}k_2\right)d^2\right)$, where X denotes the time used for message passing and aggregation, $\overline{|\mathcal{E}|}$ and $\overline{|\mathcal{P}^{sche}|}$ denote the size of positive samples for node- and schema-level tasks, k_1 and k_2 denote the number of negative samples per positive sample at the node- and schema-level, and d means the embedding dimension.

Algorithm 2 The training algorithm of PT-HGNN

Require: Heterogeneous graph $\mathcal{G}$, a GNN model f_θ

Initialize model parameters θ with Xavier initialization
Obtain compressed graph $\mathcal{G}'$ via edge sparsification
for each batch node $\mathcal{V}_B$ in $\mathcal{G}'$ **do**
  Prepare the negative samples with Eq. (1)
  Prepare the negative schema samples with Eq. (4)
  for each node u in $\mathcal{V}_B$ **do**
    Prepare the negative samples $\mathcal{N}_u^2$ with Eq. (5)
    for each relation $R \in \mathcal{R}$ **do**
      Calculate the $\mathcal{L}_{u,R}^{node}$ by Eq. (2)
    end for
    for each positive schema instance $s^+ \in \mathcal{P}_u^{sche}$ **do**
      Calculate $\mathcal{L}_u^{sche}$ by Eq. (9)
    end for
  end for
  Calculate $\mathcal{L}$ by Eq. (12)
  Update parameters θ
end for
return Pre-trained GNN model f_{θ^*} for downstream tasks

Time complexity of offline edge sparsification. The time used in the offline edge sparsification stage is linear *w.r.t.* the number of nodes in the graphs, which is given by $O\left(\frac{|\mathcal{V}|}{\epsilon\sqrt{\alpha\beta}}\right)$, in which $|V|$ denotes the number of nodes in the graph, and ϵ, α and β are parameters related to the random walk as shown in Algorithm 1. The value $\frac{1}{\epsilon\sqrt{\alpha\beta}}$ is less than 1000 in practice.

Putting it all together, the overall time complexity of the pre-processing and pre-training is $O\left(X + |\mathcal{V}|\left(|\overline{\mathcal{E}}|k_1 + \overline{|\mathcal{P}^{sche}|}k_2\right)d^2\right)$ + $O\left(\frac{|\mathcal{V}|}{\epsilon\sqrt{\alpha\beta}}\right)$. However, according to the implementation details in Section 4.1, $O\left(\left(|\overline{\mathcal{E}}|k_1 + \overline{|\mathcal{P}^{sche}|}k_2\right)d^2\right) \gg O\left(\frac{1}{\epsilon\sqrt{\alpha\beta}}\right)$. Therefore, the time overhead is mainly induced by the pre-training procedure, and the execution time of pre-processing is negligible.

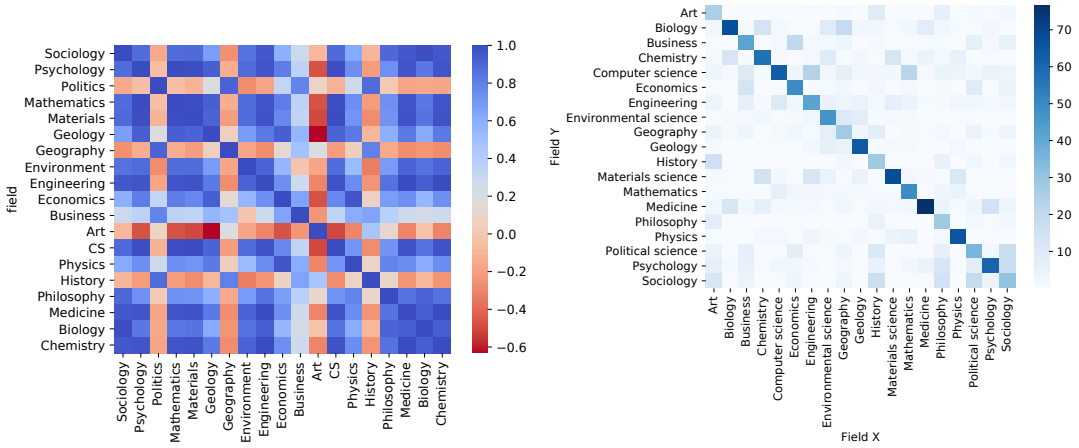
C STRUCTURAL PROPERTY OF GRAPHS

We extract eight commonly used graph property metrics [24] as shown in Table 6, each of which quantifies a specific structural property of graphs. Regarding the five graphs we used in the experiments, the Art graph has a large difference in most of the graph properties from the others. We compute the correlation matrix of different graphs using these graph properties, as shown in Figure 3(a). The correlation value between two graphs indicate the similarity of their network structures. In addition, Figure 3(b) shows the citation coefficient[1] between different fields, which is the percentage of publications in a field X that cites papers in field Y. A

[1] https://academic.microsoft.com/topics/

Table 6: Structural properties of Open Academic Graph (OAG)

Field	Density (1e-5)	Attribute assortativity	Degree assortativity(1e-2)	Degree centrality (1e-5)	Eigenvector centrality (1e-4)	Triangle num	Average clustering	Transitivity (1e-4)
CS	2.280	-0.156	-3.304	2.280	4.749	98.903	0.127	5.341
Materials	2.794	-0.184	-4.423	2.794	5.475	44.702	0.096	9.223
Engineering	1.496	-0.288	-4.395	1.496	4.689	27.424	0.100	2.849
Chemistry	1.069	-0.157	-3.676	1.069	3.236	70.976	0.112	3.197
Art	11.720	-0.465	-10.180	11.720	18.087	9.981	0.129	4.415
Sociology	4.585	-0.336	-6.504	4.585	9.0255	24.193	0.113	3.673
Psychology	3.879	-0.121	-4.102	3.878	6.399	66.911	0.112	9.583
Politics	29.626	-0.224	-8.940	29.626	25.712	13.314	0.106	32.064
Mathematics	5.380	-0.223	-5.446	5.379	8.648	57.533	0.157	9.539
Geology	6.771	-0.125	-3.445	6.770	8.134	93.157	0.127	21.458
Geography	21.299	-0.101	-6.338	21.299	19.659	9.709	0.092	45.714
Environmental	4.092	-0.305	-5.704	4.092	8.908	9.796	0.074	4.614
Economics	7.833	-0.153	-3.750	7.832	10.369	44.071	0.110	15.748
Business	17.854	-0.114	-6.000	17.854	15.967	36.176	0.100	31.982
Physics	3.433	-0.243	-2.947	3.433	18.911	82.106	0.111	12.896
History	32.909	-0.273	-9.838	32.908	28.972	10.640	0.118	20.680
Philosophy	8.671	-0.465	-9.017	8.671	14.826	14.845	0.132	4.061
Medicine	0.739	-0.177	-3.664	0.739	2.795	57.734	0.099	2.143
Biology	0.744	-0.106	-2.876	1.069	2.577	80.870	0.134	2.176



(a) Correlation matrix

(b) The percentage fields Y referencing fields X

Figure 3: Correlation matrix and citation between fields

larger citation coefficient generally means the higher similarity between two fields.

D IMPLEMENTATION OF BASE GNN MODEL AND BASELINES

We utilize Intel(R) Xeon(R) Platinum 8268 CPU and Tesla V100 to run experiments with both the pre-training and downstream tasks. In the sampling stage, we follow the settings of the HGSampling sampler in previous studies [12] for fair comparison. The parameter settings of the baseline models are as follows: (1) **EdgePred** [9] is simply used to predict whether there exists a link between two nodes; (2) **DGI** [30]: we apply mean pooling to the whole graphs for the generation of the global graph summary embeddings, and we shuffle the graphs to construct the negative sample set for conducting contrastive learning; (3) **ContextPred** [11]: we capture the 2~5-hop neighbors to generate the context embedding; (4) **GraphCL** [35]: we remove 20% of the nodes in the node dropping stage, and in the pre-training task, we utilize the 3-hop neighbors to generate subgraphs in context embedding for contrastive learning; (5) **GPT-GNN** [12]: we utilize the attribute generation and edge generation tasks for pre-training GNNs, in which the parameter settings follow the code[2].

[2]https://github.com/acbull/GPT-GNN