Published: 10 July 2025
Online AM: 05 December 2024
Accepted: 26 November 2024
Revised: 30 August 2024
Received: 26 January 2024

RESEARCH-ARTICLE

Denoising Heterogeneous Graph Pre-training Framework for Recommendation

LEI SANG, Anhui University, Hefei, Anhui, China

YU WANG, Anhui University, Hefei, Anhui, China

YIWEN ZHANG, Anhui University, Hefei, Anhui, China

XINDONG WU, Hefei University of Technology, Hefei, Anhui, China

Denoising Heterogeneous Graph Pre-training Framework for Recommendation

LEI SANG, Anhui University, Hefei, China
YU WANG, School of Computer Science and Technology, Anhui University, Hefei, China
YIWEN ZHANG, Anhui University, Hefei, China
XINDONG WU, Hefei University of Technology, Hefei, China

Heterogeneous graph neural networks (HGNN) have exhibited significant performance gains by modeling the information propagation process in graph-structured data for recommender systems. However, existing HGNN-based Recommendation still face two challenges: (1) They overlook the rich semantics brought by the combination of different meta-paths, making it difficult to capture the importance of various meta-paths; (2) when HGNN use meta-paths to capture high-order information, they are susceptible to noise data, as noise from connected nodes can create cumulative effects on a target node in the graph. To tackle these issues, we propose a new model called the **Denoising Heterogeneous Graph Pre-training Framework (DHGPF)** to enhance recommendation tasks. This framework has two stages: pre-training and training. In the pre-training stage, we assign learnable weights to different meta-paths and use a simplified multi-layer graph convolution network to automatically aggregate semantic information from different meta-path combinations. This approach can capture the importance of these paths. The training stage focuses on reducing noise using gating mechanism and denoising structure learning methods. These methods accomplish the denoising process through information filtering. Our model was evaluated on three real-world datasets, demonstrating that DHGPF outperforms other state-of-the-art recommendation methods. We have further organized the source code of the article at https://github.com/wangyu0627/DHGPF.

CCS Concepts: • **Information systems** → **Recommender systems**;

Additional Key Words and Phrases: Pre-training, Denoising Recommendation, Heterogeneous Graph Neural Networks

ACM Reference format:
Lei Sang, Yu Wang, Yiwen Zhang, and Xindong Wu. 2025. Denoising Heterogeneous Graph Pre-training Framework for Recommendation. *ACM Trans. Inf. Syst.* 43, 5, Article 121 (July 2025), 31 pages.
https://doi.org/10.1145/3706632

This work was supported by the National Science Foundation of China (No. 62272001, No. 62206004, and No. 62206002), Hefei Key Common Technology Project (No. 2023SGJ014), and Xunfei Zhiyuan Digital Transformation Innovation Research Special for Universities (No. 2023ZY001).
Authors' Contact Information: Lei Sang, Anhui University, Hefei, China; e-mail: sanglei@ahu.edu.cn; Yu Wang, School of Computer Science and Technology, Anhui University, Hefei, China; e-mail: wangyuahu@stu.ahu.edu.cn; Yiwen Zhang (corresponding author), Anhui University, Hefei, China; e-mail: zhangyiwen@ahu.edu.cn; Xindong Wu, Hefei University of Technology, Hefei, China; e-mail: xwu@hfut.edu.cn.

ACM 1558-2868/2025/7-ART121
https://doi.org/10.1145/3706632

1 Introduction

The recommender system has gained increasing importance across various facets of daily life, encompassing personalized content delivery in domains such as short videos, news, and shopping. **Collaborative filtering (CF)** stands as a pivotal technique in recommender systems, leveraging historical user–item interaction data to provide personalized recommendations based on user-user or item-item similarity. Traditional collaborative filtering methods predominantly employ matrix factorization techniques [14, 33] to represent user–item interactions as a matrix and derive latent feature vectors for users and items through matrix decomposition. Implicit feedback [15, 59] enriches user–item interactions quality by incorporating implicit user behavior information, such as click counts and browsing history. However, general recommendation approaches often neglect the auxiliary attributes of users and the labeled features of items, relying solely on user–item interaction graphs to anticipate user preferences. This reliance leads to a notable data sparsity issue.

Heterogeneous information networks (HIN) can exploit hidden rich relations in various types of auxiliary data to address the problem of data sparsity [10]. Existing HIN-based recommendation models [11, 32, 35, 48] leverage pre-defined meta-paths to generate user or item embeddings from HIN. Meta-paths describe specific sequences of node types and relationships in HIN. For example, in Figure 1(b), the relation between two movies can be inferred by (1) "movie-Director-Movie" and (2) "**Movie-Genre-Movie (MGM)**." These meta-paths capture the semantic relations of (1) Directing by the same director and (2) belonging to the same genre. Therefore, based on the intuition that connecting two movies through meta-paths represents different semantic movie relationships, we can identify potential interesting movies. This intuition helps infer user preferences based on the movies similarity, thus generating effective recommendations. However, traditional HIN-based recommendation [11, 35] relies on one-hop meta-path random walk strategies (e.g., DeepWalk [26] and Metapath2Vec [9]) to compute the similarity between users and items for recommendation, failing to capture high-order structural and complex semantic relationships within the Figure 2.

Recently, **heterogeneous graph neural network (HGNN)** have exhibited significant performance gains by modeling the information propagation process in graph-structured data for a recommendation. In HIN-based recommendation, HGNN serves as a powerful information capture tool to effectively mine higher-order interactions among users and items. Typical HGNN [36, 50, 51, 62] employs relation graphs formed by meta-paths to learn flexible and expressive representations for nodes. For example, HAN [50] uses a hierarchical attention mechanism to assign weights for different meta-paths. HeCo [51] learns complement node embeddings from one-hop neighbors and meta-path neighbors across views. These HGNN-based recommendation methods utilize iterative propagation operations to aggregate each meta-path-based node in the graph. This aggregation leverages both local and global message passing to generate embeddings. By iterative multi-layer propagation operations, we can exploit high-order interactions between users and items, generating high-quality embeddings for recommended items or users.

Despite their promising performance, HGNN-based recommendation methods encounter two significant challenges that hinder the comprehensive learning of embeddings for recommendation.

CH1: Meta-path Combination Challenge. The multiple relational graphs of users in the Movielens dataset, constructed through different meta-paths, are shown in Figure 2. From the figure, we observe that the semantic information brought by different meta-paths is diverse. HGNN-based recommendations [2, 52] typically employ diverse meta-paths to capture rich semantics, enhancing the preferences of users and items. However, adding more meta-paths would result in higher computational costs and time expenses. The pre-training framework can help us to solve the cost problem well through its powerful knowledge transfer and feature extraction. Graph-based pre-training methods [18, 19] tend to use unsupervised manner, which may not be optimal for recommendation

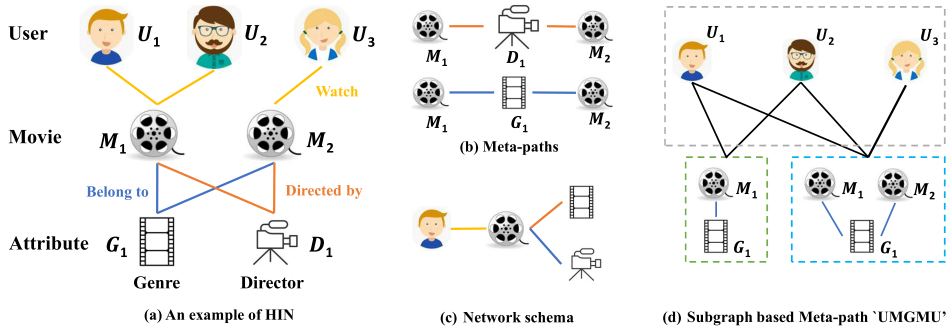
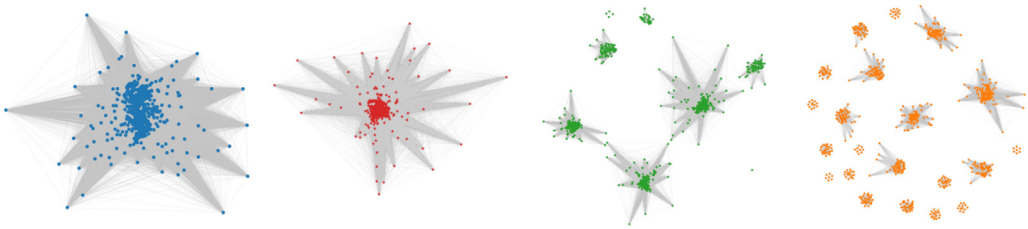


Fig. 1. A toy example of an HIN on Movie dataset.



(a) Meta-path 'UMU' (b) Meta-path 'UMGMU' (c) Meta-path 'UAU' (d) Meta-path 'UOU'

Fig. 2. Visualization of subgraphs based on different meta-paths in dataset Movielens, and different meta-paths represent different semantic information: (a) users who have interacted with the same movie; (b) users who have interacted with the same movie and have the same movie genre; (c) users belonging to the same age group; (d) users belonging to the same occupation.

task. Moreover, the heterogeneous graph-based pre-training recommendation method PreHIN4Rec [8] proposes incorporating GNN for fine-tuning within existing recommendation frameworks. CHEST [44] employs curriculum learning to capture local contextual information of subgraphs. These methods lies in using meta-paths to construct relation-subgraphs, but they only consider one meta-path to represent one semantic, thus overlooking the combination of multiple semantics. Therefore, designing a pre-training framework for heterogeneous recommendation remains a significant challenge at present.

CH2: Denoising Challenge. While the pre-training framework introducing meta-paths that bring more interactions to enrich semantic information, it also inevitably introduces noise [56, 69]. Aggregating information from noisy nodes can impair the quality of node embeddings, leading to incorrect predictions in recommendation. We argue that noise attacks [38] could potentially inflict more severe damage on the training of HGNNs than on standard neural networks due to the propagation process of HGNNs. The propagation of information in HGNNs may lead to cascading effects, whereby minor noise could exert an impact on distant nodes. Existing graph robust methods (e.g., Pro-GNN [20] and SGCN [4]) explore the impact of graph structures like low-rankness, sparsity, and feature smoothness, but they overlook the limitations of collaborative denoising between user–item interaction graphs and relation-subgraphs in heterogeneous recommendation. Consequently, the key challenge lies in leveraging automatically learned meta-path weights post pre-training to identify noisy edges and aggregate features from the most informative neighbors.

To address the two challenges mentioned above, we propose a heterogeneous recommendation pre-training framework and a denoising training model called **Denoising Heterogeneous**

Graph Pre-training Framework (DHGPF). For *CH1*, we propose a supervised multi-layer graph convolutional pre-training framework. First, addressing the cost issue associated with adding meta-path during model training, we leverage pre-training framework. This framework defines available meta-paths using different edge types and utilizes them to decompose the HIN into multiple relationship subnetworks (graphs). The pre-training framework employs multi-layer **graph convolutional networks (GCNs)** on these subnetworks to automatically capture various meta-path lengths and relationships, which can acquire richer semantic information through a broader combination of meta-paths. Secondly, to address the unsupervised limitation in existing pre-trained models, we adopt a supervised loss. The **Bayesian Personalized Ranking (BPR)** loss [28] utilizes automatically generated interactions as supervised signals for updating the parameters.

For *CH2*, we use the pre-trained results for denoising structure learning. First, we implement a gating mechanism on the pre-trained results to facilitate selective information propagation. Then, within the **denoising structural learning (DSL)**, we employ a learnable mask matrix to regulate the participation of nodes in the aggregation process. The integration of a learnable mask and attention mechanism allows the model to focus on specific nodes, thereby enhancing its capacity to discard noise information. These two techniques are designed to achieve denoising through a dual information filtering process.

We summarize the contributions of this article as follows:

- —We propose a simple yet effective pre-training framework for heterogeneous recommendation. It automatically captures diverse semantic information by applying a multi-layer simplified graph convolution message-passing mechanism on a relation graph based on meta-paths. Through this framework, we automatically learn the weights of meta-paths, distinguishing key meta-paths from others and addressing the challenge of meta-path combination in existing models.
- —We employ a method for structure learning of user and item relation graphs, using learnable binary masks and stochastic variational optimization to obtain edge sparse graphs to denoise in the graph efficiently.
- —We evaluate DHGPF on three real datasets for the Top-K recommendation task. Experimental results show that our model improved significantly in effectively extracting heterogeneous information and addressing noise compared to baseline methods.

2 Preliminaries

In this section, we introduces the fundamental knowledge of HIN from two aspects. First, specific concepts within HIN are presented, followed by a summary of the symbols appearing in the paper to assist readers in better comprehension.

2.1 Setting of HIN

HIN is a special kind of information network, which contains multiple types of objects and multiple types of edges. We set up our recommender system in the framework of the HIN, which can be defined as follows:

HIN. An HIN is defined as a graph $G = (V, E, I, R, \phi, \varphi)$, where V and E represent the sets of nodes and edges, respectively. Each node v and edge e is linked to type mapping functions $\phi : V \rightarrow I$ and $\varphi : E \rightarrow R$, where I and R denote the sets of node and edge types, with $|I| + |R| > 2$.

Network Schema. The network schema, denoted as $T_G = (I, R)$, is used to demonstrate the high-level topology of an HIG, which describes the node types and their interaction relations. It is important to highlight that T_G represents a directed graph that is constructed over object types I, with its edges representing relations drawn from R.

Table 1. Symbol and Description

Symbol	Description
G	heterogeneous information network
V, E	node set and edge set
T_G	network schema
I, R	node type set and edge type set
ρ	a meta-path
G^ρ	Meta-path-based Subgraph ρ
M, N	number of meta-paths for users and items
ρ_M, ρ_N	Adopted user and item based meta-paths
$G_u^{\rho_M}, G_i^{\rho_N}$	subgraph of user-based and item-based meta-path
α_u, β_i	user-based and item-based meta-path-aware weights
A_M, I_N	the adjacency matrices with $G_u^{\rho_M}$ and $G_i^{\rho_N}$
$\mathbf{e}_u, \mathbf{e}_i$	initialized representations of user u, item i
$\mathbf{E}_u^{(0)}, \mathbf{E}_i^{(0)}$	initialized representation sets of user u, item i
L	the layer of meta-paths hops
$\mathbf{E}_{u,i}$	the output of multi-layer graph convolution sums the embeddings
$\mathbf{R}$	user–item interaction matrix
$\mathbf{D}$	diagonal matrix
$\mathcal{N}_u, \mathcal{N}_i$	the neighbor sets of user u and item i
Z_A, Z_I	the L_0-norm of learnable binary masks under user-user view and item-item view
W	parameters for neural network
a_{ij}	the attention coefficient for edge e_{ij} across all layers
λ_A, λ_I	regularization hyperparameters that balances between data loss and edge sparsity
σ	the sigmoid function

Meta-path. In an HIN $G = (V, E, I, R, \phi, \varphi)$, a meta-path ρ is represented as $I_1 \xrightarrow{R_1} I_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} I_{l+1}$, characterizing a composite connection between v_1 and v_l. Taking Figure 1 as an instance, multiple meta-paths, like "**Movie-User-Movie (MUM)**" and "MGM" can link entities within the HIN. Commonly, different meta-paths can reveal the different inter-dependency information of two movies. The path "MUM" indicates that two movies were watched by the same user, while "MGM" illustrates that two movies are of the same genre.

Meta-path-based Subgraph. Given a node v and a meta-path ρ, the neighbors N_v^ρ of the meta-path are defined as a set of nodes in the heterogeneous graph that are connected to node v through the meta-path. The connections of N_v^ρ and v are denoted as $\mathbf{e}_v^\rho$, and the set of all edges formed by all nodes is represented as $\mathbf{E}^\rho$. Therefore, $\mathbf{E}^\rho$, along with the set of all nodes of this specific type denoted as V^ρ, forms a subgraph G^ρ. For example, in Figure 1(d), U1 and U2 are interconnected via the meta-path "U1-M1-G1-U2," while the connection between U1 and U3 is established through the meta-path "U1-M1-G1-M2-U3." The connections among users within the grey box constitute a subgraph based on the meta-path "U-M-G-M-U."

2.2 Notations

Before introducing the proposed model, we summarize the symbols used in Table 1 in this article. We present the symbols in the order of appearance for Section 2.1, "The pre-training Framework"

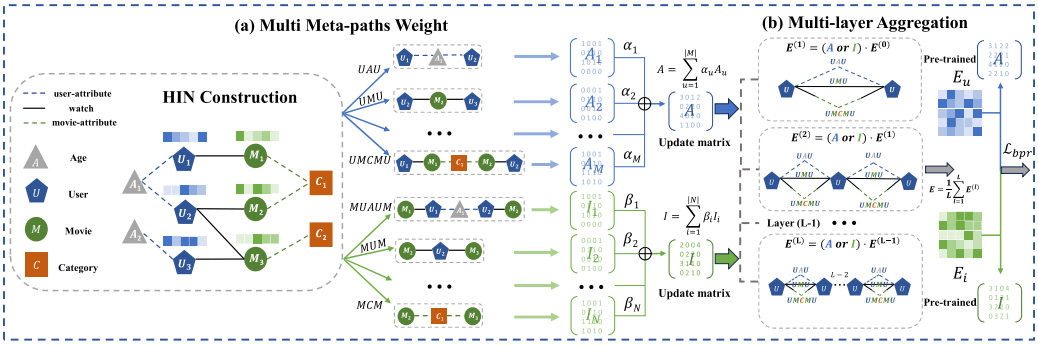


Fig. 3. The pre-training framework of our proposed DHGPF. (a) We split the HIN into individual interaction matrices for users or items using meta-paths and assign learnable weights for each matrix. (b) The framework employs multi-layer convolutional networks to capture semantic information from longer meta-paths combinations. Pre-trained weight matrices A and I are continuously updated in a supervised manner.

Section 3.1, and "The Training Model" Section 3.2 separated by lines. Users and items are denoted as u and i, respectively. If an observed interaction exists between user u and item i, it is represented as 1 in the matrix $\mathbf{R}$; otherwise, it is represented as 0.

3 Methodology

In this section, our DHGPF is divided into two stages: pre-training framework and training model. For pre-training, we use a multi-layer simplified graph convolution information propagation mechanism to automatically learn the weights of various meta-paths. The matrices formed by these weights serve as the input for the subsequent training model. Next, we utilize matrices and learnable binary masks to mitigate noise issues from specific views.

3.1 The Pre-training Framework

This section addresses the first challenge: *Meta-path combination challenge*. The pre-training framework assigns learnable weights to subgraphs based on different meta-paths, with each subgraph representing the semantics contained in the corresponding meta-path. The weight propagation process using a light GCN, and multi-layer GCN are employed to combine these semantics into varying lengths. In a supervised setting, weights are continuously updated until they eventually stabilize. We multiply the weights with the corresponding subgraph matrices to obtain a weight matrix representing individual semantics. Finally, these matrices are summed to obtain the pre-trained weight matrix used for model training.

3.1.1 Multi Meta-paths Weight. Heterogeneous graphs typically consist of multiple types of nodes and edges. We leverage this characteristic to define M meta-paths of user and N meta-paths of item respectively, where each type of meta-path ρ often represents a specific semantic, denoted as $\{\rho_1, \rho_2, \ldots, \rho_M\}$ and $\{\rho_1, \rho_2, \ldots, \rho_N\}$. Therefore, based on the "Meta-path-based Subgraph" mentioned in Section 2.1, we first generate multiple subgraphs using different types of meta-paths. As shown in Figure 3, in the recommendation scenario, we perform recommendation tasks based on the subgraphs of users and items so that we obtain *user*: $\left\{G_u^{\rho_1}, G_u^{\rho_2}, \ldots, G_u^{\rho_M}\right\}$; *item*: $\left\{G_i^{\rho_1}, G_i^{\rho_2}, \ldots, G_i^{\rho_N}\right\}$.

Next, considering that user-based and item-based meta-paths exhibit distinct dependency semantics, reflecting different preferences in their respective domains. For example, in Figure 3, M1 is connected to M2 via the meta-path "MCM," indicating the semantic that the two movies belong

to the same genre. This semantic can only indicate item domain preferences. However, it is evident that the node types "Category" and "User" are not directly connected, representing a preference solely belonging to the item domain. Therefore, in order to better capture these dependencies and preferences, our DHGPF learns user-based meta-path-aware weights as α_u: $\{\alpha_1, \alpha_2, \ldots, \alpha_M\}$, and item-based weights as β_i: $\{\beta_1, \beta_2, \ldots, \beta_N\}$. Notably, α_u and β_i are not hyperparameters but dynamically evolving trainable parameters in our recommendation task, thus residing as two independently learnable parameters throughout the model training process.

Finally, we utilize multi-layer convolution to automatically capture one-hop and high-order meta-paths with different semantic information and assign different weights to meta-paths using α_u and β_i. Unlike existing methods such as DeepWalk [26] and Metapath2vec [9], which are unsupervised pre-training methods, we use the inner product of user and item as the final representation and supervise the training with BPR [28] loss. This supervised approach helps the learned weights align better with the recommendation task. We combine the obtained subgraphs $G_u^{\rho_M}$ and $G_i^{\rho_N}$ with their adjacency matrices A_M and I_N, together with the learned α_u and β_i, obtain the final adapted matrices A and I:

$$\begin{cases} A = \sum_{u=1}^{|M|} \alpha_u A_u, & A_u \in \{A_1, A_2, \ldots, A_M\} \\ I = \sum_{i=1}^{|N|} \beta_i I_i, & I_i \in \{I_1, I_2, \ldots, I_N\} \end{cases}, \tag{1}$$

where α_u and β_i are obtained by applying weights to longer meta-path combination via multi-layer convolution. The following section will provide a detailed description of weight passing process.

3.1.2 Multi-layer Aggregation. In this section, we explain how to use multi-layer convolutions to obtain different combinations of meta-paths and learn their importance. Considering the memory usage and time cost generated when multiple layers of computation are involved, the original GCN [21] is too complex, with non-linear activation functions and learnable weight matrices. We utilize two variants to capture various meta-path combinations automatically: the simplified GCN [54], which removes non-linear activation functions, and LightGCN [13], which removes both. Here, we take the example of a user. When using only a single-layer graph convolution, the calculation of the user's embedding is as follows:

$$\begin{cases} \mathbf{E}_u^{(1)} = A \cdot \mathbf{E}_u^{(0)} \cdot W^{(1)} \\ \mathbf{E}_u^{(1)} = A \cdot \mathbf{E}_u^{(0)} \end{cases}, \tag{2}$$

where $\mathbf{E}_u^{(0)}$ are initialized by Xavier uniform distribution, and W is a learnable weight matrix. Finally, we obtain user embeddings $\mathbf{E}_u^{(1)}$ after a single-layer of GCN, and similarly, items have the same calculation process to obtain $\mathbf{E}_i^{(1)}$.

While using two GCN variants for pre-training, we found that compared to the simplified GCN, the performance of LightGCN did not decrease, and the pre-training time was significantly reduced. In the experimental section, we present detailed tables to verify this result. So, we chose it as the appropriate graph convolution function to use as the message passing function, and the embedding after the user's two-layer GCN is as follows:

$$\mathbf{E}_u^{(2)} = A \cdot \mathbf{E}_u^{(1)} = A \cdot (A \cdot \mathbf{E}_u^{(0)}) = A^2 \cdot \mathbf{E}_u^{(0)}, \tag{3}$$

here, we provide an example of a two-hop meta-path related to users, using "HIN Construction" in Figure 3 as an example of a heterogeneous graph. First, as shown in Figure 4, we define three meta-paths between two users, denoted as $U_2 \overset{\rho_1 * 0.5}{\longleftrightarrow} U_3$, $U_1 \overset{\rho_2 * 0.8}{\longleftrightarrow} U_2$, and $U_1 \overset{\rho_3 * 0.5}{\longleftrightarrow} U_2$. Corresponding to each meta-path, ρ_1, ρ_2, and ρ_3, their meta-path subgraph matrices A_1, A_2, A_3, and their weights

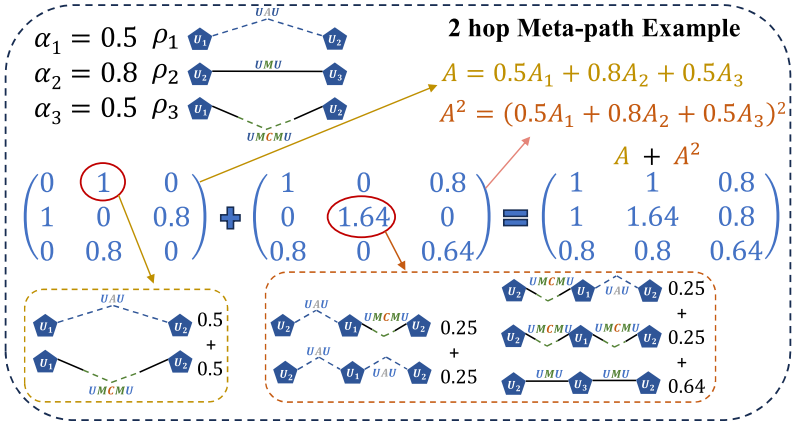


Fig. 4. A toy example of 2-hop meta-path.

$\alpha_1, \alpha_2, \alpha_3$. A is a one-hop meta-path matrix obtained by weighted summation as in Equation (1). For example, in the first matrix, $A_{(1,2)} = 1$, is obtained by weighting two one-hop meta-paths. $U_1 \xrightarrow{UAU} U_2 : 0.5$ and $U_1 \xrightarrow{UMCMU} U_2 : 0.5$. A^2 is a two-hop meta-path matrix, which can be obtained according to Equation (3). It can automatically capture the weights of random combining two-hop meta-paths. For example, in the second matrix, $A^2_{(2,2)} = 1.64$ implies five two-hop meta-paths. $U_2 \xrightarrow{UAU} U_1 \xrightarrow{UMCMU} U_2 : 0.25$, $U_2 \xrightarrow{UAU} U_1 \xrightarrow{UAU} U_2 : 0.25$, $U_2 \xrightarrow{UMCMU} U_1 \xrightarrow{UAU} U_2 : 0.25$, $U_2 \xrightarrow{UMCMU} U_1 \xrightarrow{UMCMU} U_2 : 0.25$ and $U_2 \xrightarrow{UMCMU} U_3 \xrightarrow{UMU} U_2 : 0.64$, with their weighted sum being 1.64. Finally, we fuse single-layer GCN and two-layer GCN to obtain the output $(A + A^2)$ through Equation (5). While providing an implicit supervised signal for each weight α_u through simple matrix multiplication $A^L \cdot \mathbf{E}_u^{(0)}$, allowing us to learn the meta-path weights that are aligned with the recommendation task.

To capture the longer heterogeneous meta-paths, we can extend it to L-layer. However, meta-paths longer than two hops often introduce semantic ambiguity due to their complex combinations. Considering the interference caused by longer meta-paths, we set the range for α and β to be within the interval $(0, 1)$, ensuring that longer meta-paths have lower weights.

$$\mathbf{E}_u^{(L)} = A^L \cdot \mathbf{E}_u^{(0)} = \left(\sum_{u=1}^{|M|} \alpha_u A_u \right)^L \cdot \mathbf{E}_u^{(0)}, \alpha_u \in (0, 1), \tag{4}$$

where α_u is the weight of each user-based meta-path matrix. Similarly, we obtain $\mathbf{E}_i^{(l)}$ as the embedding for items. Finally, our multi-layer graph convolution sums the embeddings obtained at each layer to obtain the output:

$$\mathbf{E}_{u,i} = \frac{1}{L} \sum_{l=1}^{L} \mathbf{E}_{u,i}^{(l)}. \tag{5}$$

3.1.3 Supervised Pre-training. To perform supervised pre-training for heterogeneous recommendation tasks, we use the BPR pairwise loss [28] function. First, we define an item that interacts with a user as the positive sample i and those that do not interact as negative sample j. We obtain the final $\mathbf{E}_u$ and $\mathbf{E}_i$, along with the corresponding embeddings $\mathbf{e}_u$ and $\mathbf{e}_i$ for each node, and compute the dot-products to get $\hat{y}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i$ and $\hat{y}_{uj} = \mathbf{e}_u^\top \mathbf{e}_j$, which represent the predicted values for the positive

and negative samples. BPR assumes that observed interactions better reflect user preferences than unobserved interactions, and thus, they should be assigned higher predicted values. The objective function is defined as follows:

$$\mathcal{L}_{bpr} = \sum_{(u,i,j)\in O} -\ln \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) + \lambda \|\mathbf{E}_u^{(0)}\|^2, \tag{6}$$

where σ denotes the sigmoid function, and λ denotes a hyperparameter to control the weight of the L_2 regularization. The final output of the pre-trained framework under the supervision of BPR loss includes weight matrices A and I, as well as embeddings $\mathbf{E}_u$ for users and $\mathbf{E}_i$ for items.

3.2 The Training Model

This section addresses the second challenge faced in the abstract: *Denoising challenge*. The training model in Figure 5, we first introduce the methods of heterogeneous information aggregation from three views of the model (i.e., user–user view, item–item view, and user–item view). We utilize the pre-trained matrices A and I as inputs for user–user view and item–item view. We then provide detailed explanations of the DSL modules for the first two views and, finally, analyze the optimization of the model's loss function.

3.2.1 Heterogeneous Information Aggregation. In model training, we divide it into three perspectives to learn embeddings for users and items separately. These three views are user-user, item-item, and user-item. First, we randomly initialize embeddings for users and items as $\mathbf{E}_u^{(0)}$ and $\mathbf{E}_i^{(0)}$, and their id-corresponding embeddings as $\mathbf{e}_u^{(0)}$, $\mathbf{e}_i^{(0)}$. We then stack them together to obtain $\mathbf{E}_{ui}^{(0)}$. Next, we apply graph convolutional neural networks on the user–item interaction graph G_{ui}. The message-passing mechanism in our DHGPF is as follows:

$$\begin{aligned} \mathbf{e}_u^{(k+1)} &= \sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \mathbf{e}_i^{(k)}, \\ \mathbf{e}_i^{(k+1)} &= \sum_{u\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \mathbf{e}_u^{(k)}, \end{aligned} \tag{7}$$

where $\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$ denotes a symmetric normalization term, and $\mathcal{N}_u$ and $\mathcal{N}_i$ are the neighbor sets of user u and item i, respectively. Inspired by LightGCN [13], in G_{ui}, we only aggregate user–item interaction without including self-connections for user-user and item-item in the convolution.

$$\mathbf{E}^{(k+1)} = \left(\mathbf{D}^{-\frac{1}{2}} \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix} \mathbf{D}^{-\frac{1}{2}}\right) \mathbf{E}^{(k)}, \tag{8}$$

where $\mathbf{R}$ denotes user–item interaction matrix, and $\mathbf{D}$ is diagonal matrix. Finally, we obtain embeddings $\mathbf{E}_u^{(3)}$ and $\mathbf{E}_i^{(3)}$ under the user-item view across 3-layer GCN.

Next, we use the pre-trained matrices A and I, and their corresponding $\mathbf{E}_u^{(0)}$ and $\mathbf{E}_i^{(0)}$ as inputs for the user-user view and item-item view, respectively. The value of each edge is determined based on the importance of capturing meta-path information in pre-training, and different values represent different combinations of semantic information. However, directly using A and I may lead to semantic confusion. Therefore, we can employ gating mechanisms to address such a problem. Gating thresholds μ_1 and μ_2 are set for A and I. An edge between two nodes is considered to exist in A if it is higher than μ_1, and similarly, for I, it corresponds to μ_2. This process yields a new user-user graph G_u and item-item graph G_i. However, it is still impossible to distinguish which edges are

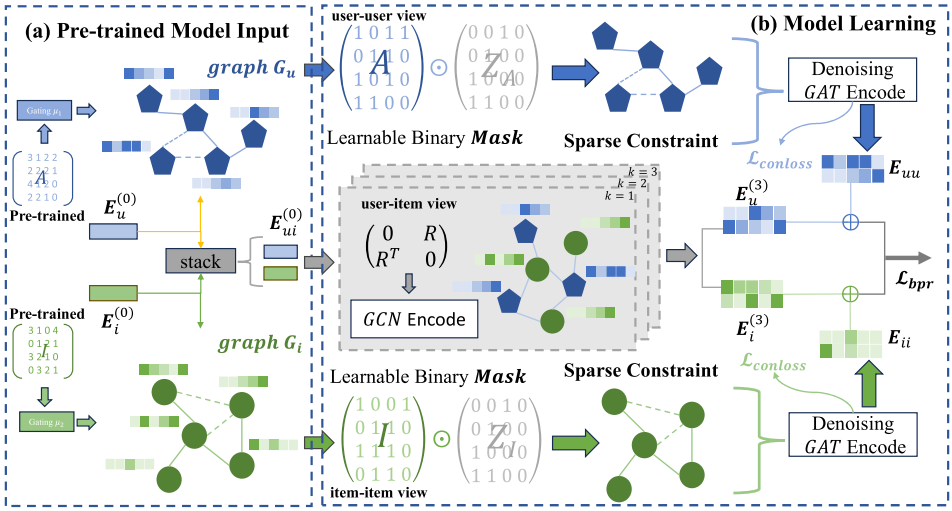


Fig. 5. The training model of our proposed DHGPF. (a) The model takes pre-trained weight matrices A and I as input, along with random initialized embeddings $\mathbf{E}_u^{(0)}$ and $\mathbf{E}_i^{(0)}$. (b) The model learning from three views, where the user-user view and item-item view utilize learnable masks for denoising.

noisy and which are useful. Therefore, we apply a denoising graph attention network to these two graphs. This part is explained in more detail in the following section.

3.2.2 DSL. To remove noisy edges, we use the binary gate z_{ij} as a gate to control whether each edge $\mathbf{e}_{ij}$ participates in neighborhood aggregation. Taking the G_u of the user-user view as an example, this is equivalent to element-wise multiplication between our matrix A and the corresponding binary mask Z_A:

$$\bar{A} = A \odot Z_A, \quad Z_A \in \{0,1\}^T, \tag{9}$$

where T is the number of edges in graph G_u. For our GAT's encoder function $f(X,(A\odot Z)_A,W)$, the objective function based on attention is as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}_i} a_{ij} h_j^{(l)} W^{(l)}\right), \tag{10}$$

where a_{ij} denotes the attention coefficient for edge e_{ij} across all layers. We calculate attention coefficients through row-wise normalization of $\bar{A}$:

$$a_{ij} = \text{normalize}\ (A_{ij}z_{ij}) = \frac{A_{ij}z_{ij}}{\sum_{k\in\mathcal{N}_i} A_{ik}z_{ik}}, \tag{11}$$

where $\mathcal{N}_i$ denotes the neighbor set of node i. Meanwhile, we set $z_{ii} = 1$ to retain self-information from self-loops in GAT. However, we omit the softmax function of GAT used for normalization because A and Z typically do not contain values less than 0.

3.2.3 Model Optimization. We describe the details of the optimization process for training model of the DHGPF. First, we obtain $\mathbf{E}_u^{(3)}$ and $\mathbf{E}_i^{(3)}$ from the user-item view. Then, we apply denoising GAT to the user-user view and item-item view to obtain embeddings for node, i.e., $\mathbf{e}_{uu}$, $\mathbf{e}_{ii}$, and their corresponding embedding sets, $\mathbf{E}_{uu}$ and $\mathbf{E}_{ii}$. Finally, we combine the embeddings from three views to obtain the final $\mathbf{E}_u$ and $\mathbf{E}_i$. The embeddings are supervised using the BPR loss in Equation (6),

providing implicit feedback signals for DSL during gradient descent. The overall loss function is as follows:

$$\mathcal{L} = \mathcal{L}_{bpr} + \lambda_A \|Z_A\|_0 + \lambda_I \|Z_I\|_0, \tag{12}$$

where Z_A and Z_I denote the L_0-norm of learnable binary masks under user-user and item-item views, respectively. λ_A and λ_I are regularization hyperparameters that balance data loss and edge sparsity. Z is a matrix consisting of only 0 and 1, in both the ($A \odot Z_A$ or $I \odot Z_I$) in the denoising GAT, neither can perform a regular gradient descent. However, in the fields of machine learning and statistics, there are multiple existing solutions to this problem. We employ the methods of Stochastic Variational Optimization [1] and The Hard Concrete Gradient Estimator [24] to optimize L_0 regularization of Z and ($A \odot Z_A$ or $I \odot Z_I$), respectively.

Because $z_{ij} \forall (i,j) \in E$ is a binary random variable, we assume z_{ij} is subject to a Bernoulli distribution with parameter $\pi_{ij} \in [0,1]$, i.e., $z_{ij} \sim \text{Ber}(z_{ij}; \pi_{ij})$.

$$\|Z\|_0 = \sum_{(i,j)\in E} \pi_{ij} = \sum_{(i,j)\in E} \sigma\left(\log \alpha_{ij} - \beta \log \frac{-\gamma}{\zeta}\right), \tag{13}$$

where α_{ij} is identical to that in Equation (10). Now Z is differentiable. We further approximate to estimate the gradient of ($A \odot Z_A$ or $I \odot Z_I$):

$$Z = g(f(\log \alpha, u)), \tag{14}$$

with

$$f(\log \alpha, u) = \sigma((\log u - \log(1-u) + \log \alpha)/\beta)(\zeta - \gamma) + \gamma$$
$$g(\cdot) = \min(1, \max(0, \cdot)),$$

where σ is the sigmoid function, and $\beta = 2/3, \gamma = -0.1$ and $\zeta = 1.1$ are the typical parameter values of the hard concrete distribution. As shown in Algorithm 1, the overall process of the training model for DHGPF is as follows.

3.3 Model Analysis

In this section, we focus on the modules of DHGPF, attempting to provide reasonable concepts or explanations. We address the problem encountered in the pre-training framework, and then explore the interpretability of structural learning for denoising in the training model.

3.3.1 The Pre-training Framework. In the update of embedding $\mathbf{E}_{u,i}$, negative values may occur, leading to a phenomenon where the weights of meta-paths have both positive and negative values. To ensure that the learned weights are not affected by updates of user and item embeddings, we use the ReLu function to ensure that the embeddings of users and items in each layer of graph convolution are always non-negative. This method allows the learned weights to reflect the importance of this meta-path. From the perspective of BPR loss function [28], this can also be explained: our top-K recommendations utilize the dot product of learned user and item embeddings to obtain the ranking results. Giving higher weight to more important meta-paths results in a larger dot product, making it easier for items interacting with users to receive recommendations.

At the same time, to ensure the coherence and understandability of the model's partial descriptions, we provide the transformation function for the learned weights here. Due to our observation that the values of the pre-trained meta-path weights vary significantly, it is necessary to map them

Algorithm 1: The Training Model of DHGPF

Input: User-item interaction matrix $\mathbf{R}$, heterogenous side information comprising user-user interaction matrix A, item-item interaction matrix I, and parameters required for the training process
Output: The user/item embeddings $\mathbf{E}^{(0)} = (\mathbf{E}_u^{(0)}, \mathbf{E}_i^{(0)})$
Randomly initialize parameters $\mathbf{E}^{(0)}$ with the default Xavier distribution;
$t = 0$;
while $t < T$ **do**
  for each view in {user-item, user-user, item-item} **do**
    if view is user-item **then**
      Acquire embeddings $\mathbf{E}_u^{(3)}$ and $\mathbf{E}_i^{(3)}$ across a 3-layer GCN by utilizing Equations (7) and (8);
    else
      Use gating threshold values μ_1 and μ_2 to filter adjacency matrices A and I, obtaining graphs G_u and G_i;
      Obtain denoised representations $\mathbf{E}_{uu}$ via adjacency matrix G_u and $\mathbf{E}_{ii}$ via adjacency matrix G_i, applying Equations (9)–(14);
    end if
  end for
  Weight $\mathbf{E}_u^{(3)}$ and $\mathbf{E}_i^{(3)}$, along with $\mathbf{E}_{uu}$ and $\mathbf{E}_{ii}$ respectively, to derive the final embeddings $\mathbf{E}_u$ and $\mathbf{E}_i$;
  Calculate BRP loss $\mathcal{L}_{bpr}$ with Equation (6);
  Calculate the loss with sparse constraints using Equations (13) and (14);
  Calculate the joint learning loss $\mathcal{L}$ using Equation (12);
  Backpropagation and update parameters $\mathbf{E}$ with $\mathcal{L}$;
end while
return $\mathbf{E}_u$ and $\mathbf{E}_i$;

to the interval (0, 1). The transformation function is as follows:

$$\begin{cases} 2\ \text{sigmoid}\ (x) - 1, & x \geqslant 0 \\ 0.01x + 0.2, & x < 0 \end{cases},$$

where x denotes the weight of each meta-path. We obtain a distribution of x values that is between −5 and 10. Such a function provides appropriate discrimination, as it maps the weights that negatively impact the recommendation task to smaller values, while allocating larger values to those beneficial for recommendations.

In contrast to the bipartite graphs of general Recommendation, each type of meta-path can be regarded as a semantic information channel in heterogeneous graphs. An effective meta-path encoder is necessary for capturing relationships' heterogeneity and reusability. Specifically, we first decouple the HIN into multiple relational subgraphs through different meta-paths and then reaggregate the subgraphs. The aggregation process utilizes light graph convolution to capture meta-path information spanning various semantics lengths effectively, exploring these meta-paths' importance (weights) in heterogeneous graph representation learning.

3.3.2 The Training Model. First, we conduct the recommendation task from multiple perspectives, dividing it into user, item, and user-item views. In recent years, the rapid development of

Table 2. Comparisons of Theoretical Time Complexity

Model		Adjacency Matrix	GCN Encoder	Loss
Pre-training	HAN [50]	$O(M \cdot m^2 + N \cdot n^2)$	$O\left(d \cdot \left(Mm^2 + Nn^2\right) + d^2 \cdot (Mm + Nn)\right)$	$O(2Od)$
	PreHIN4Rec [8]		$O\left(Ld \cdot \left(Mm^2 + Nn^2\right) + Ld^2 \cdot (Mm + Nn)\right)$	$O(Od)$
	DHGPF (Ours)		$O\left(Ld \cdot \left(Mm^2 + Nn^2\right)\right)$	$O(2Od)$
Training	LightGCN [13]	$O((m+n)^2))$	$O\left(Ld \cdot (m+n)^2\right)$	$O(2Od)$
	SGL [55]		$O\left((1+2\rho)Ld \cdot (m+n)^2\right)$	$O(3Od + OSd)$
	HGCL [6]	$O((m+n)^2) + m^2 + n^2)$	$O\left(Ld \cdot (m+n)^2 + d \cdot (Mm^2 + Nn^2)\right)$	$O(3Od + OSd)$
	DHGPF (Ours)		$O\left(Ld \cdot (m+n)^2 + d \cdot (m^2 + n^2)\right)$	$O(2Od)$

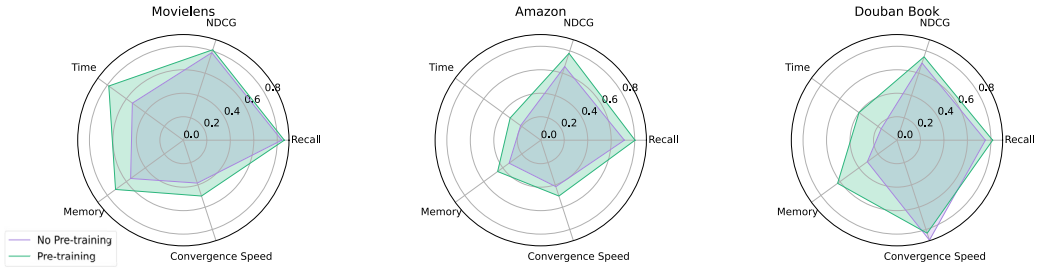
general recommendation has brought about significant improvements in performance. However, there are noise issues [56, 69] present in HINs. Therefore, heterogeneous information serves more as side information to improve the performance of the main task (user-item view). Second, we utilize a learnable masked denoising module to filter and denoise pre-trained mixed semantics within two perspectives: user-user view and item-item view.

Graph structure learning [25, 29, 58] has successfully applied in **graph neural networks (GNNs)**, capable of distinguishing intricate relationships, capturing subtle patterns, and extracting meaningful representations from graph-structured data. Because user-user and item-item graphs in the real world typically exhibit fewer interactions, we incorporate sparse and low-rank constraints into the attention mechanism to further reduce edge redundancy and remove noise edges.

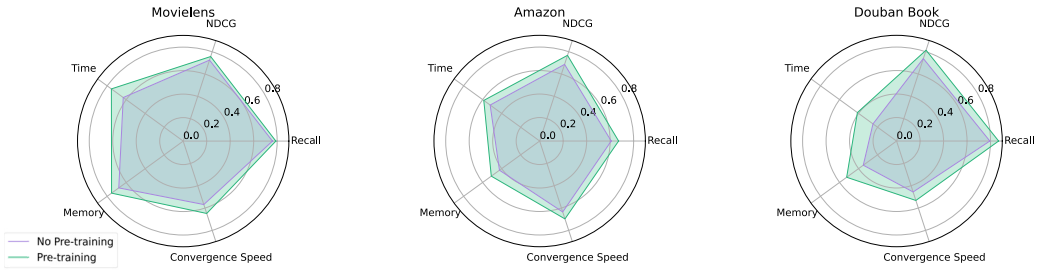
Finally, leveraging the learned edge-sparse graph for recommendation tasks can mitigate the risk of model overfitting and provide a certain level of robustness against noise attacks. Compared to information loss caused by auto-encoders [57] and the absence of edge-sparse graphs in DropEdge [29], our denoising structure learning permanently removes noisy edges in each iteration to maintain the consistency of the learned graph.

3.3.3 Time Complexity Analyses. We theoretically analyze the time complexity of DHGPF in detail, dividing it into pre-training and training stages. First, the computational complexity of our pre-training framework mainly depends on the number of meta-paths. It is known that the number of users is m and the number of items is n. There are M user matrices $\mathrm{R}^{m \times m}$ and N item matrices $\mathrm{R}^{n \times n}$ for simplified multi-layer graph convolution calculations. Therefore, the time complexity of the pre-training phase is $O\left(Ld \cdot \left(Mm^2 + Nn^2\right)\right)$, where L is the number of layers and d is the embedding dimensions. Next, we analyze the training stage of the framework. From the user-item view, the model uses LightGCN [13] to address the complexity of the bipartite interaction graph, which is $O\left(Ld \cdot (m+n)^2\right)$. From the user-user view, the complexity of the initial threshold filtering stage is the processing of each element in the matrix, with a maximum of $O\left(m^2\right)$. Next, the attention mechanism calculation is $O\left(d \cdot m^2\right)$, and the reparameterization technique is $O\left(m^2\right)$. Finally, since the embedding dimension d is much greater than 2, the time complexity of the training phase is $O\left(Ld \cdot (m+n)^2 + d \cdot m^2 + d \cdot n^2\right)$.

We compare the DHGPF's pre-training stage with HAN and PreHIN4Rec, and the training stage with LightGCN and HGCL. As shown in Table 2, we observe that in the pre-training phase, the input of the model is uniformly M user-based subgraphs and N item-based subgraphs. However, in the encoder module, due to the composite multi-relational GCN architecture of PreHIN4Rec, its complexity is much higher than that of HAN and the proposed DHGPF. Our pre-training model's complexity is slightly higher than HAN's, as it employs a multi-layer graph network to automatically combine multi-hop meta-paths, while HAN can only capture one-hop meta-path information. Finally, from the results of the complexity comparison of training models, our DHGPF is slightly higher than LightGCN but lower than the self-supervised model (i.e., SGL and HGCL). SGL constructs multiple views using graph structure perturbation, while HGCL encodes subgraphs



(a) The HGCL model utilizes pre-training to generate radar charts wrt. evaluation indicators



(b) The SMIN model utilizes pre-training to generate radar charts wrt. evaluation indicators

Fig. 6. The pre-training framework shows improvements over several baseline models wrt. Recall, NDCG, Time, Memory, and Convergence Speed.

based on meta-paths, and these processes require additional encoding time. As for the role of the pre-training framework in reducing computational costs and time expenses, this is demonstrated in Figure 6. The pre-trained weight matrix effectively reduces the baseline's training time, memory usage, and convergence speed.

4 Experiments

4.1 Experiments Setup

This section provides a comprehensive overview of the experimental setup, encompassing the datasets used, baseline models, implementations, and evaluation metrics. The experimental setup is a pivotal component in ensuring the reproducibility and accuracy of our research.

4.1.1 Datasets. In our experiments, we use three publicly available real-world datasets from different domains: Movielens[1] for movie recommendations, Amazon[2] for shopping recommendations, and Douban Book[3] for book recommendations. These datasets vary in size and sparsity, with an increasing trend. We present the details of each dataset in Table 3.

In every dataset, the training set comprises randomly chosen 80% of a user's historical interactions, with the test set encompassing the remaining interactions. To create the training set, 10% of interactions are sampled randomly for validation purposes, aiding in hyperparameter tuning. Observed user–item interactions constitute positive samples, while negative samples are crafted by pairing each positive interaction with an item previously unobserved by the user, following a sampling strategy.

[1] https://grouplens.org/datasets/movielens/
[2] http://jmcauley.ucsd.edu/data/amazon/
[3] https://book.douban.com/

Table 3. Statistics of Datasets Used in Experiments

Dataset (Sparsity)	Relations (A-B)	Number of A	Number of B	Number of (A-B)	Ave. degrees of A	Ave. degrees of B	Meta-paths
Movielens-1M (95.81%)	User-Movie	6,040	3,952	1,000,209	165.6	253.1	UMU, MUM
	User-User	6,040	6,040	302,000	50	50	UAU, MM
	User-Age	6,040	7	6,040	1.0	862.9	UOU, MGM
	User-Occupation	6,040	21	6,040	1.0	287.6	UU, MUUUM
	Movie-Genre	3,952	18	6,408	1.6	356.0	UMMMU
	Movie-Movie	3,952	3,952	79,040	20	20	UMGMU
Amazon (98.85%)	User-Item	6,170	2,753	195,791	31.7	71.1	UIU, IUI
	Item-View	2,753	3,857	5,694	2.1	1.5	UIBIU, IVI
	Item-Category	2,753	22	5,508	2.0	250.4	UICIU, ICI
	Item-Brand	2,753	334	2,753	1.0	8.2	IBI
Douban Book (99.73%)	User-Book	13,024	22,347	792,062	60.8	35.4	
	User-Group	13,024	2,936	1,189,271	91.3	405.1	UBU, BUB
	User-User	13,024	13,024	169,150	13.0	13.0	UGU, BAB
	User-Location	13,024	38	10,592	0.8	278.7	ULU, BPB
	Book-Author	22,347	10,805	21,907	1.0	2.0	UU, BYB
	Book-Publisher	22,347	1,815	21,773	1.0	12.0	UBABU, BUUUB
	Book-Year	22,347	64	21,192	0.9	331.1	

4.1.2 Baselines. To evaluate the performance of our DHGPF, we compare it with several state-of-the-art models, which serve as baselines. Details of each baseline model are presented below.

—*HERec* [35] is a recommendation model leveraging HIN via meta-path-based random walks for node embeddings, integrating fusion functions and extended matrix factorization, demonstrating improved recommendation performance and effectiveness in addressing the cold-start problem.
—*KGAT* [48] utilizes **Knowledge Graphs (KGs)** to capture intricate item relations and attributes, outperforming traditional approaches by recursively refining node embeddings with attention mechanisms, offering enhanced accuracy and interpretability in recommendations.
—*NGCF+* [49] enhances recommender systems by incorporating user–item interactions through a bipartite graph structure. We extend it to perform graph convolution-based message passing and aggregation on heterogeneous graphs.
—*HAN* [50] effectively captures node-to-neighbor importance and meta-path significance by utilizing node-level and semantic-level attention mechanisms. We use it to learn the embeddings of user-based and item-based meta-paths for the recommendation dataset.
—*HGT* [17] introduces a novel architecture for handling heterogeneous structures in Web-scale graphs by incorporating node-type and edge-type dependent parameters for attention, temporal encoding for dynamic graphs, and a specialized mini-batch graph sampling algorithm (HGSampling) for efficient training.
—*LightGCN* [13] a classic work in graph collaborative filtering, which removes the feature transformation and nonlinear activations of GCN to achieve light recommendation.
—*HeCo* [51] is a self-supervised method for HGNN using cross-view contrastive learning. It utilizes network schema and meta-path views to learn node embeddings, enabling collaborative supervision between views.
—*SMIN* [23] enhances existing social recommender systems by integrating social and knowledge-aware structures through a meta-path-guided HGNNs. It leverages self-supervised graph-based collaborative filtering to capture diverse user-item dependencies.

— *RoHe* [63] is a novel robust HGNN framework against adversarial attacks, and its proposed attention purifier can eliminate malicious neighbors and address soft attention mechanism issues based on topology and feature information.
— *PreHIN4Rec* [8] converts the HIN into a multi-relational graph, learning the representations of user and item nodes through a multi-relational GCN and fusing these representations into a unified embedding space.
— *HGCL* [6] integrates heterogeneous relational semantics by leveraging contrastive self-supervised learning and adaptive contrastive augmentation and adapts to interactions using meta networks.

4.1.3 Implementations. Our baseline models primarily focus on utilizing original code resources from the original creators (if a pytorch version is available) or using OpenHGNN [12]. OpenHGNN stands as a comprehensive and unified open-source platform tailored for the construction and replication of diverse algorithms within HIN.

Several of these models were initially integrated for tasks centered on classifying nodes within heterogeneous graphs (such as HAN, HeCo, HGT). Within this framework, we uniformly replace their loss functions with the BPR [66] loss while maintaining the fundamental structure unaltered. We standardize the dimensions of the embedding and training batches to 128 and 2048, respectively, ensuring an equitable comparison. Employing the Adam optimizer as a universal model optimizer, we initialize the parameters of each model following the Xavier distribution. Fine-tuning of parameters takes place on the validation set, accompanied by implementing an early stopping strategy to prevent potential model overfitting. The evaluation hinges on comparing recall metrics for each model, with training termination occurring upon 20 consecutive epochs without a discernible increase.

We rely on employing the most effective parameters outlined in their respective papers (if available) for the baseline methods. After this, we introduce minor modifications to discern performance trends, culminating in selecting optimal outcomes tailored to our experimental framework.

4.1.4 Metrics. Each model calculates preference scores for all items related to each user in the test set, excluding the positive samples from the training set. The top-K recommendation approach is employed, which involves recommending K items to the test users for evaluation. Evaluation metrics [67] such as recall@K and NDCG@K are used, with K set to 20.

Recall@k measures the proportion of accurately projected positive items to the total number of samples. For a specific user u, these metrics can be determined using the following formulas:

$$\text{Recall@}K = \frac{1}{M} \sum_{u \in \mathcal{U}} \frac{|\mathcal{R}_{\text{K}}(u) \cap \mathcal{T}(u)|}{\mathcal{T}(u)},$$

where $\mathcal{R}_{\text{K}}(u)$ is the top-K recommendation list based on the training set, and $\mathcal{T}(u)$ is the list of user's ground truth on the testing set.

The NDCG@K Metric centers on the ranked position list of recommended items:

$$\text{NDCG@}K = \frac{\text{DCG @}K}{\text{IDCG @}K},$$

where DCG is the Discounted Cumulative Gain, which considers the ranking order factor: this approach assigns higher gains to items ranked at the top and applies diminishing returns to lower-ranked items. IDCG is the ideal DCG, which arranges the results in the best state and calculates

Table 4. Performance Comparison of the Pre-training Framework

	Method	Movielens-1M			Amazon			Douban-Book		
		Recall	**NDCG**	**time/s**	**Recall**	**NDCG**	**time/s**	**Recall**	**NDCG**	**time/s**
1-Layer	SGCN	0.2321	0.3582	249.6	0.1333	0.0936	18.92	0.1208	0.1114	495.3
	LGCN	0.2313	0.3580	**238.7**	0.1331	0.0916	**18.45**	0.1242	0.1140	**474.5**
2-Layer	SGCN	0.2347	0.3626	284.8	0.1366	0.0938	22.07	0.1276	0.1175	553.8
	LGCN	**0.2386**	**0.3679**	266.8	**0.1399**	**0.0979**	21.04	**0.1345**	**0.1204**	528.7
3-Layer	SGCN	0.2320	0.3569	337.2	0.1345	0.0919	24.75	0.1215	0.1138	632.7
	LGCN	0.2335	0.3592	306.9	0.1352	0.0941	23.60	0.1270	0.1168	592.6
4-Layer	SGCN	0.2293	0.3497	384.3	0.1290	0.0885	28.87	0.1199	0.1087	712.6
	LGCN	0.2301	0.3522	359.0	0.1321	0.0923	26.23	0.1243	0.1101	658.1

"time" represents the time required for training each epoch, bold font indicates the best performance and shorter time, and underline indicates the second-best.

the DCG of the query under this arrangement. These are defined as follows:

$$\text{DCG } @K = \sum_{i=1}^{K} \frac{2r_i - 1}{\log_2(i+1)},$$

$$\text{IDCG@K} = K = \sum_{i=1}^{K} \frac{1}{\log_2(i+1)},$$

where r_i represents the level of relevance at the ith position and can generally be processed with 0 or 1. If the item at this position is in the test set, $r_i = 1$; otherwise, it is 0. The NDCG values of i are all within the range of 0–1.

4.2 Performance Comparison

To substantiate the effectiveness and credibility of our model, our analysis of its performance is divided into two parts: performance of the pre-training framework and performance of the training model enhanced by pre-trained results. These investigations further clarify our motivations.

4.2.1 Performance of the Pre-training Framework. In this section, we explore the performance of pre-training under supervision for recommendations, which is crucial for the reliability and credibility of the framework. Suppose the performance does not have advantages compared to other models. In that case, it cannot highlight the effectiveness of our pre-training, especially models like HAN [50] and HERec [35] that also consider the weight of meta-paths. We make the following analysis based on the phenomena observed from Table 4:

—First, as mentioned in Section 3.1, SGCN represents a simplified GCN without linear transformation, while LGCN denotes LightGCN. We use an n-layer GCN to capture meta-path combinations with a maximum length of N-hop. It can be observed that the optimal performance is achieved when the number of GCN layers is 2. The possible reason is that one layer ignores the rich semantic information from multi-hop meta-path combinations. In contrast, more layers introduce irrelevant information and noise, leading to performance degradation. Then, from the perspective of training time, we keep the GPU and other parameters

Table 5. Performance Comparison of Different Methods Using the Original Datasets

Dataset	Movielens-1M				Amazon				Douban Book			
Metrics	R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20
(2018)HERec	0.1454	0.3623	0.2313	0.3504	0.0814	0.0766	0.1355	0.0983	0.0978	0.1228	0.1374	0.1268
(2019)KGAT	0.1480	0.3697	0.2326	0.3519	0.0802	0.0750	0.1281	0.0918	0.0903	0.1154	0.1321	0.1208
(2019)NGCF+	0.1521	0.3795	0.2350	0.3602	0.0758	0.0661	0.1203	0.0802	0.0792	0.0994	0.1200	0.1062
(2019)HAN	0.1341	0.3560	0.2137	0.3386	0.0748	0.0681	0.1206	0.0816	0.0909	0.1173	0.1235	0.1130
(2020)HGT	0.1382	0.3601	0.2247	0.3440	0.0686	0.0623	0.1137	0.0781	0.0736	0.0856	0.1167	0.1054
(2020)LightGCN	0.1617	0.3920	0.2540	0.3791	0.0978	0.0926	0.1522	0.1101	0.0968	0.1312	0.1471	0.1370
(2021)HeCo	0.1506	0.3804	0.2334	0.3624	0.0630	0.0584	0.1065	0.0726	0.0968	0.1236	0.1395	0.1300
(2021)SMIN	0.1531	0.3838	0.2388	0.3676	0.0724	0.0634	0.1218	0.0824	0.0840	0.1071	0.1271	0.1181
(2022)RoHe	0.1543	0.3832	0.2418	0.3690	0.0784	0.0720	0.1348	0.0918	0.0868	0.1110	0.1303	0.1121
(2022)PreHIN4Rec	0.1594	0.3846	0.2533	0.3760	0.0987	0.0935	0.1530	0.1104	0.1024	0.1356	0.1531	0.1418
(2023)HGCL	0.1599	0.3802	0.2521	0.3707	0.0890	0.0813	0.1424	0.0991	0.1036	0.1305	0.1506	0.1382
(ours)DHGPF	**0.1668**	**0.4219**	**0.2609**	**0.3967**	**0.1091**	**0.1032**	**0.1647**	**0.1201**	**0.1231**	**0.1551**	**0.1762**	**0.1635**
Improv.(%)	3.15%	7.63%	2.72%	4.64%	10.54%	10.37%	7.65%	8.79%	18.82%	18.22%	17.00%	18.31%

The best results are shown in bold, and the second-best results are underlined. The "Improve" shows the relative improvement of DHGPF over the second-best results. Our DHGPF outperforms the existing baselines.

unchanged. Although fewer layers mean shorter training time, considering the performance improvement, we set the number of pre-training layers to 2 after weighing the pros and cons.

—Finally, we compare with the baseline models in Table 5 of the original graph under the condition of top-$K = 20$. When more meta-paths are added, our DHGPF achieved good recommendation performance in all three datasets. We found during experiments that the performance of existing models is not better with more meta-paths added. On the contrary, some meta-paths can lead to performance degradation. Our model learn negative weights for these meta-paths, reducing their impact on pre-training performance. Therefore, we used manually selected meta-paths in our subsequent experiments to achieve optimal performance for most models. This further demonstrates that our pre-training is generic and straightforward, eliminating manually selecting meta-paths, thereby automatically capturing their importance (weight) in the recommendation.

4.2.2 Performance of the Training Model. Table 5 reports the recommendation performance of all baseline models on three public datasets. Based on the results of the evaluation metrics, we summarize possible conjectures or explanations for these outcomes.

—Our DHGPF achieves the best recommendation performance on three datasets. Compared to the suboptimal methods, our model shows a general improvement of around 10% to 20%, with the most significant improvement seen in the Douban Book dataset (with a relative increase of up to 18.82%), which is a substantial advancement. The commonality in our approach is that we learn embeddings from three perspectives: user, item, and user-item for recommendation tasks. The difference is that HGCL uses contrastive learning to enhance the fusion of embeddings from different perspectives, while we employ a pre-training and denoising graph structure learning approach. The performance improvement demonstrates the effectiveness of our method.

—The methods based on HGNNs (such as Rohe [63], HeCo [51], HGT [17]) generally offer better performance compared to other alternative methods (such as KGAT [48], NGCF+ [49]). This phenomenon indicates the compatibility of heterogeneous knowledge networks enriched with social and item information with recommender systems. Additionally, contrastive or self-supervised learning methods (such as HGCL [6], HeCo [51], SMIN [23]) also demonstrate

Table 6. Performance Comparison Using the Datasets with 10% Noisy User–Item Interactions

Dataset	Movielens-1M				Amazon				Douban Book			
Metrics	R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20	R@10	N@10	R@20	N@20
(2018)HERec	0.1454	0.3623	0.2313	0.3504	0.0814	0.0766	0.1355	0.0983	0.0978	0.1228	0.1374	0.1268
(2019)KGAT	0.1480	0.3697	0.2326	0.3519	0.0802	0.0750	0.1281	0.0918	0.0903	0.1154	0.1321	0.1208
(2019)NGCF+	0.1521	0.3795	0.2350	0.3602	0.0758	0.0661	0.1203	0.0802	0.0792	0.0994	0.1200	0.1062
(2019)HAN	0.1341	0.3560	0.2137	0.3386	0.0748	0.0681	0.1206	0.0816	0.0909	0.1173	0.1235	0.1130
(2020)HGT	0.1382	0.3601	0.2247	0.3440	0.0686	0.0623	0.1137	0.0781	0.0736	0.0856	0.1167	0.1054
(2020)LightGCN	0.1617	0.3920	0.2540	0.3791	0.0978	0.0926	0.1522	0.1101	0.0968	0.1312	0.1471	0.1370
(2021)HeCo	0.1506	0.3804	0.2334	0.3624	0.0630	0.0584	0.1065	0.0726	0.0968	0.1236	0.1395	0.1300
(2021)SMIN	0.1531	0.3838	0.2388	0.3676	0.0724	0.0634	0.1218	0.0824	0.0840	0.1071	0.1271	0.1181
(2022)RoHe	0.1543	0.3832	0.2418	0.3690	0.0784	0.0720	0.1348	0.0918	0.0868	0.1110	0.1303	0.1121
(2022)PreHIN4Rec	0.1594	0.3846	0.2533	0.3760	0.0987	0.0935	0.1530	0.1104	0.1024	0.1356	0.1531	0.1418
(2023)HGCL	0.1599	0.3802	0.2521	0.3707	0.0890	0.0813	0.1424	0.0991	0.1036	0.1305	0.1506	0.1382
(ours)DHGPF	**0.1668**	**0.4219**	**0.2609**	**0.3967**	**0.1091**	**0.1032**	**0.1647**	**0.1201**	**0.1231**	**0.1551**	**0.1762**	**0.1635**
Improv.(%)	3.15%	7.63%	2.72%	4.64%	10.54%	10.37%	7.65%	8.79%	18.82%	18.22%	17.00%	18.31%

The best results are shown in bold and the second-best results are underlined. The "Improve" denotes the relative improvement of DHGPF over the second-best results. Our DHGPF outperforms the existing competitors. In all cases, our DHGPF achieves a higher relative improvement over the original datasets.

significant potential, highlighting that contrastive learning techniques effectively enhance the rationality of user–item interaction encoding and personalized embeddings.

—HAN [50] and HERec [35], as two of the earlier methods for exploring meta-path weights, achieved decent recommendation performance. However, they both have certain limitations. Possible reasons are that, compared to other methods, they still use manual settings for meta-path selection, ignoring the rich semantic information generated by various meta-path combinations. In the recommendation task, unlike HAN, which uses semantic attention to automatically assign weights to different meta-paths, it is difficult to provide enough discrimination using the softmax function. Our DHGPF effectively addresses this issue. Although HERec is designed to integrate functions to extract information from meta-paths, it does not utilize GCNs to learn information from meta-paths of different lengths and combinations.

—LightGCN [13] is a classic graph recommendation method, and its performance on relatively dense datasets remains powerful. Although PreHIN4Rec [8] uses LightGCN as the base model for enhancement, its improvement is quite effective. Our overall framework also uses LightGCN as the main task, achieving a significant improvement in accuracy. This phenomenon indicates that a rich combination of meta-paths is particularly important for capturing hidden preferences in recommendations.

—Table 6 presents a comparative analysis of the performance of various models under a 10% random noise attack. It is observable that the introduction of noise interactions precipitated a decline in recommendation performance across all models, with the trend being more pronounced on sparser datasets. This trend can be attributed to the fact that lesser supervisory signals often lead to amplified negative impacts from erroneous feedback. On comparing the results presented in both tables, it is evident that our proposed model, DHGPF, consistently outperforms the suboptimal models across all scenarios. Both HGCL and DHGPF, being graph-based recommendation models, primarily experience performance degradation due to the magnification of negative impacts from noise interactions by the message-passing mechanism inherent in GNN. However, we could effectively mitigate the influence of noise by employing pre-training and denoising techniques.

—In general, the most important means of extracting information from HGNN is to define new interactions through meta-paths to obtain rich semantic information. Therefore, by combining the characteristics of recommender systems, our pre-trained method effectively explores the

importance of different semantic information to provide high-quality interactions for the training part. The performance gap between our DHGPF and methods that also use meta-paths confirms that adaptive learning weights for meta-paths enhance performance by simplifying GCN. Furthermore, DSL, using sparsity and variational optimization, further enhances the accuracy of information. This can be observed through Section 9.

4.3 Pre-training Framework for Baselines Models

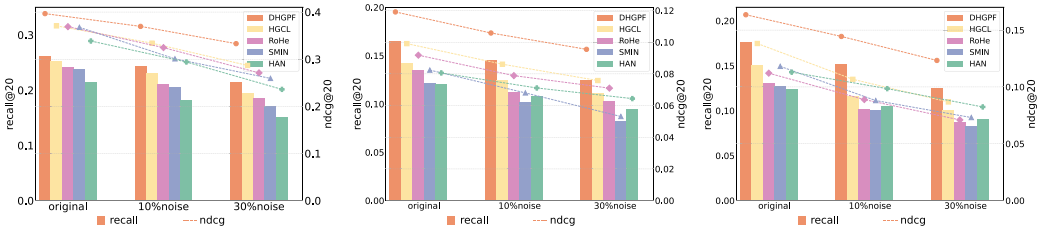
Figure 6 illustrates the enhancements of two baseline models that directly apply the pre-training framework across multiple dimensions. The radar chart is partitioned into five metrics, each quantified by a value between 0 and 1. Firstly, recommendation performance metrics such as Recall and NDCG are proportionately scaled within this range. Secondly, computational efficiency metrics like Time, Memory, and Convergence Speed are inversely beneficial, meaning lower values signify better performance; thus, we take their reciprocal and scale it to compute the actual values.

—Following the use of the relation graph Gu and Gi obtained from pre-training, HGCL has achieved significant performance improvement on all three datasets. However, the primary aspect is not the improvement per se but the crucial efficiency gains during model training and fine-tuning. The only drawback is a slightly slower convergence speed on the Douban Book dataset than before pre-training. We hypothesize that this reflects the trade-off for better performance, though the advantages in storage and time outweigh this minor shortcoming.
—SMIN exhibits a similar trend to HGCL. However, due to its more complex model structure, the improvements in various aspects are less pronounced than those in HGCL. Particularly on large datasets that closely resemble real-world conditions, pre-training techniques provide a wealth of semantic information for us to filter, proving highly beneficial for general recommendation tasks involving heterogeneous information fusion.
—The series of figures demonstrates the universality and effectiveness of the pre-training component of our DHGPF. One primary reason is that DHGPF undergoes pre-training under the rich semantics brought by more meta-paths and captures the importance of each meta-path. The larger scale and simpler model allow the semantic knowledge to transfer to downstream training tasks, better perceiving the relationships between users and items.

4.4 Denoising Structure Learning for Noisy Interactions

In this section, we delve into the denoising capability of the model. First, we randomly add a portion of user–item interactions without introducing any new users or items. The specific approach is to select items, denoted as item j, in the training dataset that user u from interaction $\langle u, i \rangle$ has never interacted with, creating noisy edges $\langle u, j \rangle$. Secondly, we preserve other heterogeneous information undisturbed to construct a new heterogeneous graph. Our DHGPF and the compared models rely on undisturbed heterogeneous information to enhance recommendation performance. Finally, we have selected representative models to compare recommendation performance under these two conditions, as shown in Figure 7, and provide explanations or hypotheses for the experimental observations.

—Most models exhibit an obvious decreasing trend in noise ranging from 10% to 30%. In contrast, our DHGPF shows an approximately linear decreasing trend, even with the second segment of the drop rate on the Amazon dataset being lower than the first. HGCL introduces noise from multiple perspectives despite its contrastive learning-enhanced knowledge transfer across different views. This noise inevitably leads to a significant decrease in performance. Notably, HAN demonstrates an excellent ability to handle noise. This can be attributed to its process of reducing the impact of user-item noise through the fusion of indistinguishable meta-paths.



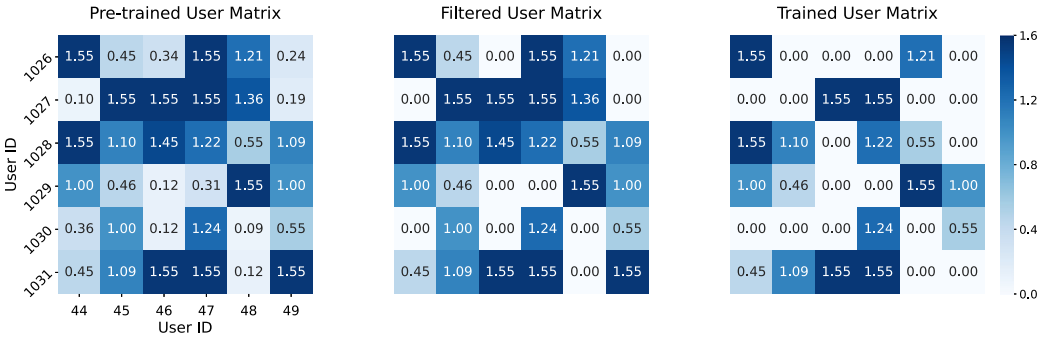
(a) Movielens (b) Amazon (c) Douban Book

Fig. 7. Performance under increasing noise interaction. We introduce 30% random noise and 10% noise for comparison to explore the ability of DHGPF against noise further. On all three datasets, the histogram represents recall@20, and the line graphs represent NDCG@20. The combined analysis of the declining trends in both metrics demonstrates the model's robustness.

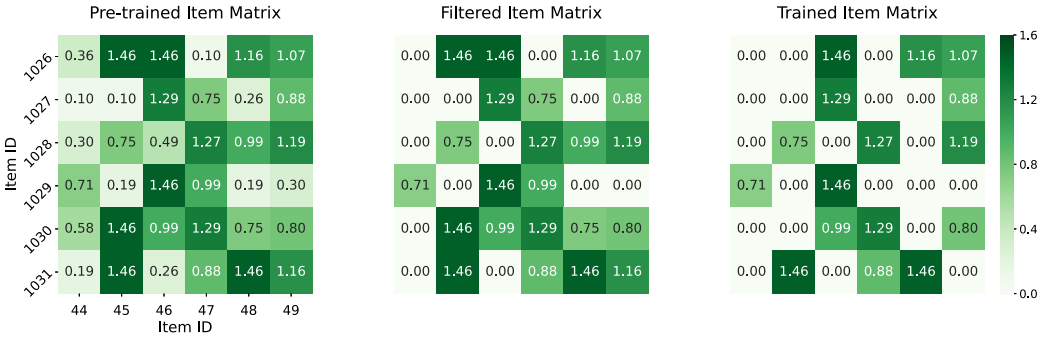
Table 7. Performance Comparison of Denoising Module

Datasets	Noise Rate	Metrics	DHGPF-GAT	DHGPF	Imp.%	
Movielens	0%	Recall	0.2558	0.2609	**1.99**	-
		NDCG	0.3900	0.3967	**1.72**	-
	10%	Recall	0.2345	0.2441	**4.09**	↑
		NDCG	0.3554	0.3696	**4.00**	↑
	30%	Recall	0.2034	0.2148	**5.60**	↑
		NDCG	0.3159	0.3330	**5.41**	↑
Amazon	0%	Recall	0.1603	0.1647	**2.74**	-
		NDCG	0.1179	0.1201	**1.87**	-
	10%	Recall	0.1413	0.1453	**2.83**	↑
		NDCG	0.0998	0.1058	**6.01**	↑
	30%	Recall	0.1187	0.1251	**5.39**	↑
		NDCG	0.0894	0.0955	**6.82**	↑
Douban Book	0%	Recall	0.1730	0.1762	**1.85**	-
		NDCG	0.1600	0.1635	**2.19**	-
	10%	Recall	0.1471	0.1517	**3.13**	↑
		NDCG	0.1397	0.1444	**3.36**	↑
	30%	Recall	0.1186	0.1252	**5.56**	↑
		NDCG	0.1169	0.1233	**5.47**	↑

—In Table 7, the bold is the relative improvement rate. DHGPF-GAT denotes the model that utilizes a standard attention mechanism module, while DHGPF signifies the training process that employs a denoising attention mechanism. By taking the scenario without noise as the baseline, we observe that as the noise ratio incrementally escalates, the relative improvement rate of DHGPF's evaluation metrics also experiences a continuous surge. This phenomenon indicates that our denoising module exhibits superior robustness when confronting additional noise interference. This can be attributed to the learnable mask matrix's ability to filter out noise edges to a certain extent and exclude them from participating in the testing phase.



(a) Weight matrix for users in the training stage.



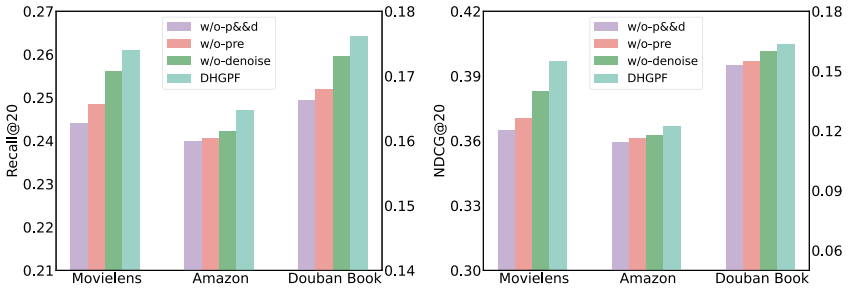
(b) Weight matrix for items in the training stage.

Fig. 8. Case study on Douban Book dataset. *Pre-trained matrix*: The matrix obtained by accumulating weights from different meta-paths after pre-training. *Filtered matrix*: The weight matrix filtered by a threshold through a gating mechanism. *Trained matrix*: The final matrix obtained from the previous matrix through a DSL network.

4.5 Case Study

We present the case study of user-user and item-item weight matrices during the training stage in Figure 8. *We describe the case through the following steps*:

- —Firstly, we extract a 6×6 matrix from the complete pre-trained matrix. For example, the value between user 1026 and user 47 is 1.55, indicating that these two users are connected by five meta-paths: "UBU" (0.999), "UGU" (0.097), "ULU" (0.090), "UU" (0.243), and "UBABU" (0.121). To facilitate observation, we chose the same IDs in the Item matrix.
- —Secondly, we filter out interactions below the thresholds determined from parameter analysis: $\mu_1 = 0.4$ for users and $\mu_2 = 0.6$ for items, thus obtaining the first reconstructed graph. It can be observed that user 1026 and user 47 still have interactions, with this process only removing a small number of edges (i.e., cells with weights set to 0). However, increasing such thresholds would make it difficult for the denoising module to function effectively.
- —Finally, we obtain the trained matrix by putting the final embeddings into the denoising module. We can observe that, compared to the filtered matrix, the DSL tends to retain interactions with larger values from the pre-trained matrix. Considering that a larger value between two users indicates richer semantic relations, the DSL strategy aligns with the idea that users with closer connections are more likely to purchase the same items. However, it is undeniable that DSL removes a large number of useless interactions in this process, significantly improving



(a) Recall score (b) NDCG score

Fig. 9. Experimental results of ablation study. The axes on the left belong to the Movielens dataset and the axes on the right belong to other datasets.

the model's performance. For example, the largest value between user 1026 and user 47 is removed at this stage.

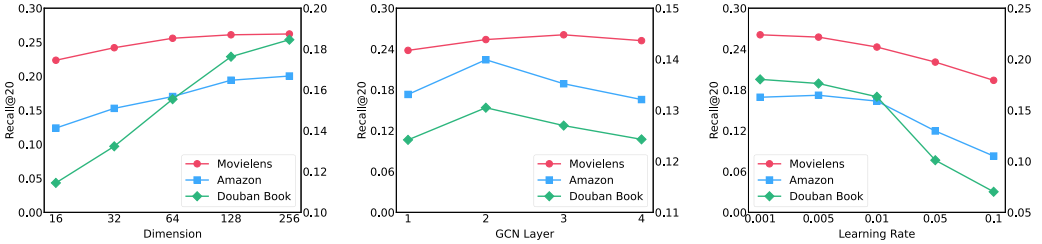
The above process describes how DHGPF employs sparse and low-rank attention mechanisms to discover and downgrade the noisy effect. In summary, graph structure learning [25, 57] has been successful in the field of GNNs, capturing complex relations in graph data to learn useful interactions. DHGPF's denoising module permanently removes noisy edges in each iteration to maintain the consistency of the learned graph. Such edge-sparse graphs for recommendation tasks can mitigate the risk of model overfitting while providing a certain degree of robustness against noise attacks.

4.6 Ablation Study

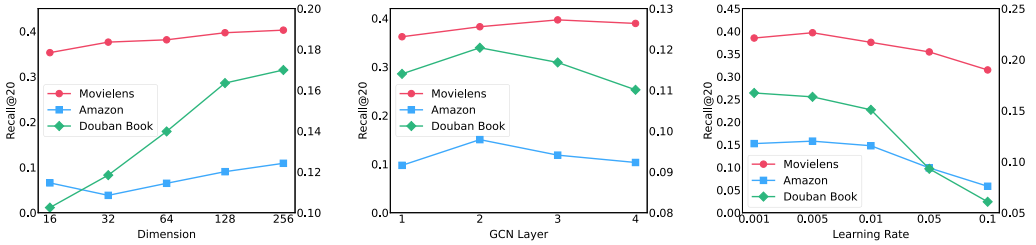
We conducted ablation experiments to validate the effectiveness of key components in our DHGPF and made possible conjectures or explanations regarding the experimental results. Here is our explanation of variants for several models.

- —***w*/*o*−pre**: This is a variant that cancels the pre-training framework. We set the weight of each meta-path to 1, meaning both α_u and β_i in Equation (1) are set to 1. We weigh the user and item matrices, filter them with a gating mechanism, and obtain the final matrix. We train it using a denoising training model to obtain experimental results.
- —***w*/*o*−denoise**: This variant eliminates the DSL part of the user-user and item-item views. We directly use the matrices learned during pre-training and train them with GAT [42] for multi-view embedding fusion to obtain experimental results.
- —***w*/*o*−p&&d**: The third variant does not have the two modules mentioned above; it simply allows the GAT to obtain embeddings in noisy user-user and item-item views and then merge them with the user-item view.

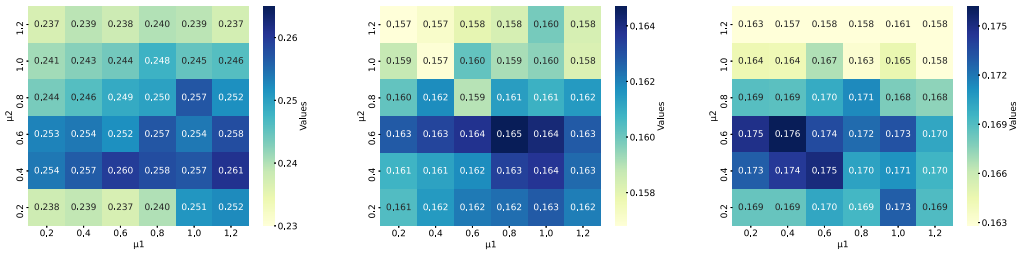
In Figure 9, the performance metrics for the MovieLens dataset are represented on the left axis, while for Amazon and Douban Book, they are on the right axis. It can be observed that, in all cases, removing the pre-training "***w*/*o*−pre**" consistently results in worse performance compared to other variants. Additionally, variants where both pre-training and denoising "***w*/*o*−p&&d**" are removed generally exhibit the poorest performance in most cases. This further highlights the effectiveness of utilizing graph convolutions to capture meta-paths' importance and how learning denoising graph structures helps us aggregate crucial information more effectively.



(a) Recall score w.r.t. dimension d, pre-training layers L and learning rate λ.



(b) NDCG score w.r.t. dimension d, pre-training layers L and learning rate λ.



(c) Recall score w.r.t. gating thresholds $\mu 1$ and $\mu 2$. From left to right, movielens, amazon, and douban book.

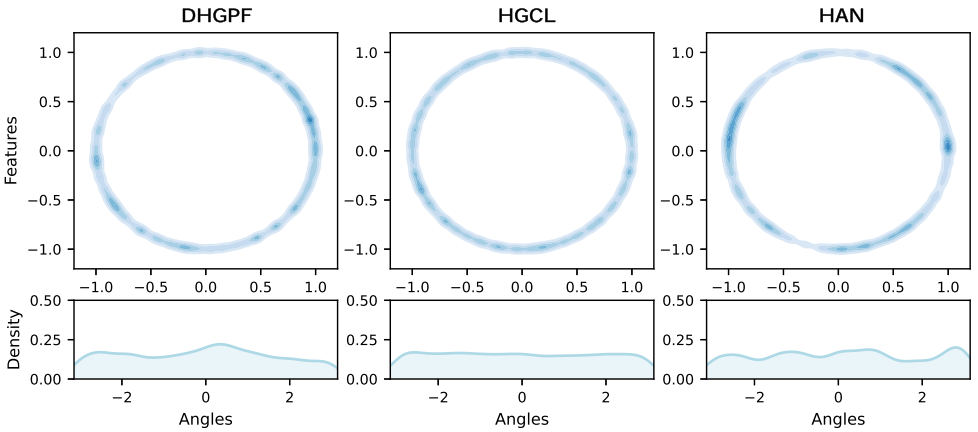
Fig. 10. Parameter sensitivity of proposed method w.r.t. pre-training layers, dimension d, learning rate λ and gating thresholds. The axes on the left belong to the Movielens dataset and the axes on the right belong to other datasets.

4.7 Hyperparameter Analysis

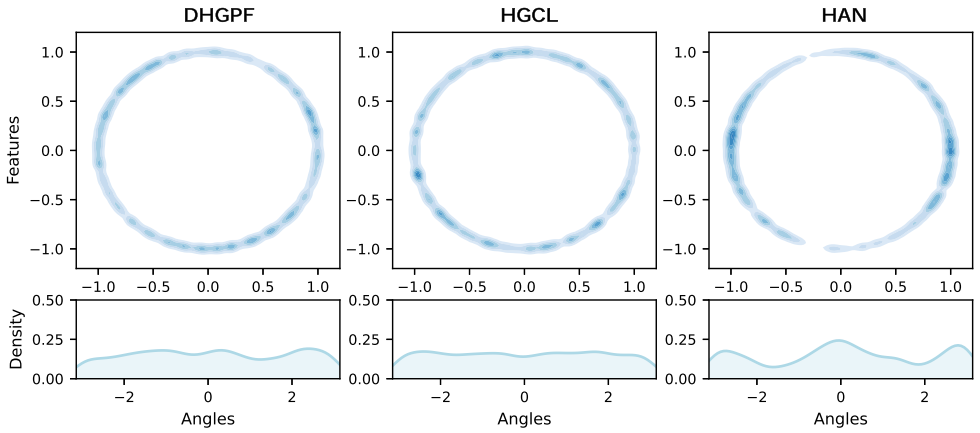
In this study, we comprehensively explored key hyperparameters' impacts on heterogeneous graph-based recommender systems. We investigated the effects of embedding dimensions, GCN layer variations in pre-training, gating thresholds for multi-view optimization, and learning rates.

— *Embedding dimension.* We explored the impact of embedding dimensions on the performance of nodes in a heterogeneous graph, ranging from 16 to 256. In Figure 10, we observed that, except for the Movielens dataset, which achieved the highest performance at a dimension of 128, other datasets continued to improve their performance at a dimension of 256. However, considering the diminishing rate of improvement and the increased training time and memory cost associated with larger dimensions, we decided to fix the dimension at 128 for all experiments. Additionally, due to the risk of overfitting in GNNs, the increase in dimension is limited.

— *GCN layers of pre-training.* In the pre-training process, we continuously increase the number of layers of GCN to capture information from nodes that are further away through different meta-path combinations. As shown in Figure 10, with the increase in the number of layers, the recommendation performance peaks at the second layer and then starts to decline. This is



(a) Distribution of user representations learned from the dataset of Douban Book.



(b) Distribution of item representations learned from the dataset of Douban Book.

Fig. 11. We plot feature distributions with Gaussian kernel density estimation in $\mathbf{R}^2$ (the darker the color is, the more points fall in that area.) and KDE on angles (i.e., $\arctan 2(y, x)$ for each point $(x, y) \in \mathcal{S}^1$).

because one-hop and two-hop meta-paths are already effective in capturing the topology of the heterogeneous graph for recommendation tasks, and further convolutions introduce more noise, leading to a decrease in performance.

— *Gating thresholds of multi-view.* In Section 3.2.1, we set gating thresholds $\mu 1$ and $\mu 2$, for the pre-trained matrices A and I to determine whether the edges in the graph undergo training in the next phase. As shown in Figure 10(c), we optimize these two hyperparameters using an adaptive optimization method to achieve the best performance, and then combine the two optimal parameters to obtain the final experimental results.

— *Learning rate.* To investigate the impact of learning rate, we conduct comparative experiments in Figure 10 by using different learning rate settings (0.001, 0.005, 0.01, 0.05, 0.1). The optimal learning rate is used for training on different datasets, and other contrast methods are kept under the same optimal learning rate.

In general, we observed that while different dimensions affected performance across datasets, the optimal dimension was set at 128 due to diminishing returns. Additionally, increasing GCN layers

beyond the second layer reduced performance due to noise introduction. Optimizing thresholds and learning rates significantly impacted system performance, demonstrating the importance of these hyperparameters in heterogeneous graph-based recommendation models.

4.8 Representations Visualization

We observed distinct features/density distributions in the Douban Book dataset using the visualization tool [45] employed in SimGCL [61]. In the Figure 11, we first map the learned representations (randomly sample 2,000 users for each dataset) to 2-dimensional normalized vectors on the unit hypersphere $\mathcal{S}^1$ (i.e., circle with radius 1) by using t-SNE [40]. When these methods achieve the best performance, they obtain all the representations. For a clearer presentation, the density estimations on angles for each point on $\mathcal{S}^1$ are also visualized.

On the far right, HAN [50] shows noticeable breakpoints and clustering phenomena along the curves. As a classic HGNN model, HAN was initially adapted for tasks such as node classification and clustering, supervised by BPR loss. It still tends to favor a large number of interactions with popular items. In the middle is HGCL [6], where the curve is continuous, and cluster points are evenly distributed throughout the entire region. A plausible explanation is that it utilizes the latest contrastive learning techniques to obtain a more uniform distribution of representations. This technique achieves uniformity by pushing connected nodes away from high-degree hubs in the representation space. However, excessive pursuit of consistency brought by contrastive learning, while neglecting the impact of user–item interactions, may lead to performance decline. On the left is our DHGPF, positioned between these two states. It neither exhibits excessively uniform distribution nor leans towards clustering like HAN. We attribute this phenomenon to the effectiveness of heterogeneous information captured by the pre-training and denoising framework. Fusing these embeddings with those obtained from the user-item encoder allows clustering and uniformity to coexist, resulting in better recommendation performance, as demonstrated in SimGCL.

4.9 Model Fairness Analyses

Bias in recommendation systems is a critical issue [5], especially in sensitive applications involving fairness and equity [68]. DHGPF integrates various mechanisms during the pre-training and training stages to mitigate this risk. The following are the key strategies in our framework to ensure fairness and reduce bias:

- *Adaptive learning of meta-path weights*: While DHGPF uses meta-paths to construct relation-subgraphs, the model does not rely on fixed, potentially biased initial weights but instead dynamically learns these weights during pre-training. This approach allows DHGPF adjust the importance of different meta-paths based on the data, thereby reducing bias that may arise from overemphasizing certain paths related to sensitive attributes (e.g., gender, age).
- *Denoising mechanism*: DHGPF can selectively reduce the impact of certain interactions during the aggregation process by integrating a learnable mask matrix. The denoising mechanism helps eliminate or reduce the impact of user–item interactions that may be overrepresented due to inherent biases in data collection. The final recommendations are based on a more balanced and representative perspective of the data, thereby enhancing fairness.
- *Multi-view learning*: DHGPF integrates multiple perspectives (user-user, item-item, and user-item) to construct a more comprehensive data representation. This integration allows the model to cross-validate information obtained from different interaction types, helping to identify

and compensate for biases present in a single view. This leads to more fair recommendation results across different user groups.

Given the potential bias concerns in recommendation systems, expanding the discussion on how this framework might be deployed in real-world settings would be helpful. Additionally, exploring its applicability across industries such as healthcare and finance could highlight its broader impact and contextual relevance.

5 Related Work

This section focus on three key areas: HIN-based recommender systems, heterogeneous graph representation learning, and pre-training methods for GNNs. These areas are of significant importance in addressing data sparsity in recommender systems, the complexity of heterogeneous networks, and the enhancement of GNN performance.

5.1 HIN-based Recommender Systems

Recommender systems are effective information filtering methods that provide personalized content to users based on their potential interests. SGL [55] generates contrast views by edge dropout to aid contrastive learning to enhance recommendation, and NCL [22] considers the relationship between neighbor nodes to enhance collaborative filtering. However, traditional collaborative filtering algorithms [37, 46, 47] rely solely on user–item interaction data, which leads to issues of data sparsity and cold start problems, making recommendations less effective.

In recent years, recommender systems based on HIN [31, 32] have successfully addressed the challenge of modeling various types of heterogeneous side information along with user interaction behavior. This information alleviates data sparsity and cold start problems and significantly enhances the interpretability of recommender systems, leading to widespread attention and application. Furthermore, recommender systems based on heterogeneous graph representation learning demonstrate significant advantages in acquiring neighborhood information, which can be categorized into relationship-based and meta-path-based approaches. Relationship-based models such as RGCN [34] and GHCF [3] learn vector representations of relationships and utilize relationship-aware GNNs for recommendation tasks. Meta-path-based models such as MEIRec [11] and HGCL [6] leverage meta-paths to extract substructures from heterogeneous networks, capturing rich semantic information embedded in the paths. Recent works like HGCL [6] and SMIN [23] combine two approaches, relationship and meta-path, for multi-view learning. They utilize the relationship graph formed by direct user–item interactions and meta-paths for recommendation, leveraging contrastive learning techniques to enhance personalized knowledge transformation. In this work, we have designed a pre-training strategy that learns the weights of meta-paths to extract effective side information from the relationship graph between users and items, integrating it into the user-item relationship perspective. This method contributes to our model's more effective utilization of HIN for recommendation more effectively.

5.2 Heterogeneous Graph Representation

In recent years, many researchers have been modeling networked data with various types of nodes and edges as HIN. Traditional methods [21, 42] struggle to address the specificity of heterogeneous networks, which involves the heterogeneity of nodes and edges and the challenge of integrating rich information within such networks. With the rapid development of GNNs, heterogeneous graph representation learning has emerged as the most reasonable approach to tackle these issues.

Specifically, HAN [50] introduces an HGNN based on a hierarchical attention mechanism. On the other hand, HetGNN [62] relies on random walk sampling to obtain fixed-sized sets of heterogeneous neighbors. These neighbors are grouped and encoded according to node types,

and the final node representations are obtained through inter-group information fusion. Additionally, HGT introduced the popular model "transformer" [41] for the first time from the field of **Natural Language Processing (NLP)**, incorporating node-type and edge-type dependent parameters for attention mechanisms, relative temporal encoding for dynamic structural dependencies, and a specialized heterogeneous mini-batch graph sampling algorithm for efficient training. Recently, the widespread application of contrastive learning in GNNs has also inspired the field of heterogeneous graphs. HeCo [51] is a self-supervised method that leverages embedded multi-perspective contrastive learning to consider one-hop and higher-order neighborhood relationships. HeGCL [36] utilizes contrastive learning with different views (meta-path and outline) to learn effective node representations in heterogeneous graphs. Our model is also an HGNN pre-training framework based on representation learning. However, we have made optimizations and improvements specifically tailored for recommendation tasks. DHGPF utilizes structural learning for denoising recommendations, aiding the model in better extracting heterogeneous side information.

5.3 Graph Pre-training

Graph pre-training methods have gained traction, drawing inspiration from the successes seen in pre-training approaches within computer vision [30] and NLP [7, 64, 65] domains. These techniques have recently been adapted to graph datasets to enhance GNN [16, 53]. The primary objective behind pre-training GNN is to acquire model parameters that generate generalized graph representations, which can later be fine-tuned for various downstream tasks. This approach has shown promise in addressing challenges such as limited labeled data and out-of-distribution prediction.

In recent years, mutual information maximization [39] has been used in GNNs to learn representations of nodes or graphs, aiding in capturing important features and correlations among nodes or within the graph structure. DGI [43] learns unsupervised node representations in graphs by maximizing information between patches and graph summaries without relying on random walk objectives. HDGI [27] focuses on heterogeneous graph representation learning by utilizing information theory-based approaches, meta-path structures, and semantic-level attention, showcasing superior performance in unsupervised graph-related tasks like classification and clustering. Furthermore, mutual information measures the correlation between variables, while contrastive learning leverages the differences between positive and negative samples to learn more distinctive feature representations. GraphGCL [60] proposed a contrastive learning framework for graph representation learning, aiming to enhance the effectiveness of contrastive learning through data augmentation. In general, most of these methods aim to learn general node representations based on the entire graph through supervised or unsupervised means.

Moving further into the heterogeneous graph recommendation scenario, PreHIN4Rec [8] introduces heterogeneous relation-subgraphs based on user-to-user and item-to-item meta-paths, and proposes a GCN to iteratively propagate over such subgraphs to obtain multi-relation node representations. CHEST [44] introduces heterogeneous subgraphs based on user-to-item meta-paths and proposes a transformer network to capture rich semantics on such subgraphs. Our work involves utilizing BPR loss [28] for recommendation tasks to train weights of different meta-paths. We use LightGCN [13] to capture the influence of high-order combinations of different meta-paths on recommendation tasks. Fixing the parameters of the weights assists in fine-tuning the model in the subsequent training phase to enhance performance. The parameter settings during the pre-training process are straightforward, yet the volume of pre-training data is extremely large, which aligns with the pattern observed in pre-trained models.

6 Conclusion and Future Work

In this article, we focus on exploring the rich semantic information brought about by diverse meta-paths in heterogeneous recommendation. To effectively utilize these meta-paths, we propose a novel model called the DHGPF. The model has two phases: pre-training and training. The pre-training for simple yet effective heterogeneous recommendation employs a simplified multi-layer graph convolutional information propagation mechanism to address the challenge of meta-path combinations in existing models. The training phase utilizes learnable binary masks and stochastic variational optimization to obtain edge-sparse graphs, thereby mitigating the impact of noise. Our experiments on three real-world datasets demonstrate that the performance of our DHGPF is optimal. In-depth experiments further explore the superiority of our model's meta-path fusion and noise reduction capabilities. Future work aims to enhance user personalization and reduce popularity bias by leveraging heterogeneous information, further advancing the integration of heterogeneous graph techniques with recommender systems.

Heterogeneous recommendation combining pre-training and contrastive learning. We hope to use the pre-trained weight matrix to generate multiple views in future research, using a denoising learning module on these views to train user and item embeddings. Finally, we sum these embeddings and the main view embeddings for contrastive learning to extract consistent information. These contents are updated in future work as follow-up attempts worth exploring.

References

[1] Thomas Bird, Julius Kunze, and David Barber. 2018. Stochastic variational optimization. arXiv:1809.04855. Retrieved from https://arxiv.org/abs/1809.04855

[2] Desheng Cai, Shengsheng Qian, Quan Fang, Jun Hu, Wenkui Ding, and Changsheng Xu. 2022. Heterogeneous graph contrastive learning network for personalized micro-video recommendation. *IEEE Transactions on Multimedia* 25 (2022), 2761–2773.

[3] Chong Chen, Weizhi Ma, Min Zhang, Zhaowei Wang, Xiuqiang He, Chenyang Wang, Yiqun Liu, and Shaoping Ma. 2021a. Graph heterogeneous multi-relational recommendation. In *AAAI*, Vol. 35, 3958–3966.

[4] Huiyuan Chen, Lan Wang, Yusan Lin, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. 2021b. Structured graph convolutional networks with stochastic masks for recommender systems. In *SIGIR*, 614–623.

[5] Lei Chen, Le Wu, Kun Zhang, Richang Hong, Defu Lian, Zhiqiang Zhang, Jun Zhou, and Meng Wang. 2023. Improving recommendation fairness via data augmentation. In *WWW*, 1012–1020.

[6] Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, Yong Xu, and Ronghua Luo. 2023. Heterogeneous graph contrastive learning for recommendation. In *WSDM*, 544–552.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805. Retrieved from https://arxiv.org/abs/1810.04805

[8] Phuc Do and Phu Pham. 2022. Heterogeneous graph convolutional network pre-training as side information for improving recommendation. *Neural Computing and Applications* 34, 18 (2022), 15945–15961.

[9] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. Metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, 135–144.

[10] Yuxiao Dong, Ziniu Hu, Kuansan Wang, Yizhou Sun, and Jie Tang. 2020. Heterogeneous network representation learning. In *IJCAI*, Vol. 20, 4861–4867.

[11] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided heterogeneous graph neural network for intent recommendation. In *KDD*, 2478–2486.

[12] Hui Han, Tianyu Zhao, Cheng Yang, Hongyi Zhang, Yaoqi Liu, Xiao Wang, and Chuan Shi. 2022. OpenHGNN: An open source toolkit for heterogeneous graph neural network. In *CIKM*, 3993–3997.

[13] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, 639–648.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*, 173–182.

[15] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*, IEEE, 263–272.

[16] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020b. GPT-GNN: Generative pre-training of graph neural networks. In *KDD*, 1857–1867.

[17] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW*, 2704–2710.
[18] Xunqiang Jiang, Tianrui Jia, Yuan Fang, Chuan Shi, Zhe Lin, and Hui Wang. 2021. Pre-training on large-scale heterogeneous graph. In *KDD*, 756–766.
[19] Xunqiang Jiang, Yuanfu Lu, Yuan Fang, and Chuan Shi. 2021. Contrastive pre-training of GNNs on heterogeneous graphs. In *CIKM*, 803–812.
[20] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *KDD*, 66–74.
[21] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907. Retrieved from https://arxiv.org/abs/1609.02907
[22] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *WWW*, 2320–2329.
[23] Xiaoling Long, Chao Huang, Yong Xu, Huance Xu, Peng Dai, Lianghao Xia, and Liefeng Bo. 2021. Social recommendation with Self-supervised metagraph informax network. In *CIKM*, 1160–1169.
[24] Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through L_0 regularization. arXiv:1712.01312. Retrieved from https://arxiv.org/abs/1712.01312
[25] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. 2021. Learning to drop: Robust graph neural network via topological denoising. In *WSDM*, 779–787.
[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*, 701–710.
[27] Yuxiang Ren, Bo Liu, Chao Huang, Peng Dai, Liefeng Bo, and Jiawei Zhang. 2019. Heterogeneous deep graph infomax. In *AAAI*.
[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. arXiv:1205.2618. Retrieved from https://arxiv.org/abs/1205.2618
[29] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Dropedge: Towards deep graph convolutional networks on node classification. arXiv:1907.10903. Retrieved from https://arxiv.org/abs/1907.10903
[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115 (2015), 211–252.
[31] Lei Sang, Yu Wang, Yi Zhang, Yiwen Zhang, and Xindong Wu. 2024. Intent-guided heterogeneous graph contrastive learning for recommendation. arXiv:2407.17234. Retrieved from https://arxiv.org/abs/2407.17234
[32] Lei Sang, Min Xu, Shengsheng Qian, Matt Martin, Peter Li, and Xindong Wu. 2020. Context-dependent propagating-based video recommendation in multimodal heterogeneous information networks. *IEEE Transactions on Multimedia* 23 (2020), 2019–2032.
[33] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*, 285–295.
[34] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*. Springer, 593–607.
[35] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S. Yu Philip. 2018. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2018), 357–370.
[36] Gen Shi, Yifan Zhu, Jian K. Liu, and Xuesong Li. 2023. HeGCL: Advance Self-supervised learning in heterogeneous graph-level representation. *IEEE Transactions on Neural Networks and Learning Systems* 35, 10 (2023), 13914–13925.
[37] Xiaoyu Shi, Quanliang Liu, Hong Xie, Di Wu, Bo Peng, MingSheng Shang, and Defu Lian. 2023. Relieving popularity bias in interactive recommendation: A diversity-novelty-aware reinforcement learning approach. *ACM Transactions on Information Systems* 42, 2 (2023), 1–30.
[38] Lichao Sun, Yingtong Dou, Carl Yang, Kai Zhang, Ji Wang, S. Yu Philip, Lifang He, and Bo Li. 2022. Adversarial attack and defense on graph data: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 8 (2022), 7693–7711.
[39] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. 2019. On mutual information maximization for representation learning. arXiv:1907.13625. Retrieved from https://arxiv.org/abs/1907.13625
[40] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008).
[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, 6000–6010.
[42] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *STAT* 1050, 20 (2017), 10–48550.
[43] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2018. Deep graph infomax. arXiv:1809.10341. Retrieved from https://arxiv.org/abs/1809.10341

[44] Hui Wang, Kun Zhou, Xin Zhao, Jingyuan Wang, and Ji-Rong Wen. 2023. Curriculum Pre-training heterogeneous subgraph transformer for top-n recommendation. *ACM Transactions on Information Systems* 41, 1 (2023), 1–28.
[45] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*. PMLR, 9929–9939.
[46] Wenjie Wang, Fuli Feng, Xiangnan He, Xiang Wang, and Tat-Seng Chua. 2021. Deconfounded recommendation for alleviating Bias amplification. In *KDD*, 1717–1725.
[47] Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, Min Lin, and Tat-Seng Chua. 2022. Causal representation learning for out-of-distribution recommendation. In *WWW*, 3562–3571.
[48] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *KDD*, 950–958.
[49] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*, 165–174.
[50] 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.
[51] Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. 2021. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *KDD*, 1726–1736.
[52] Yifan Wang, Suyao Tang, Yuntong Lei, Weiping Song, Sheng Wang, and Ming Zhang. 2020. Disenhan: Disentangled heterogeneous graph attention network for recommendation. In *CIKM*, 1605–1614.
[53] Chenwang Wu, Defu Lian, Yong Ge, Zhihao Zhu, and Enhong Chen. 2023. Influence-driven data poisoning for robust recommender systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 10 (2023), 11915–11931.
[54] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *ICML*. PMLR, 6861–6871.
[55] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *SIGIR*, 726–735.
[56] Cheng Yang, Xumeng Gong, Chuan Shi, and Philip Yu. 2023. A post-training framework for improving heterogeneous graph neural networks. In *WWW*, 251–262.
[57] Haibo Ye, Xinjie Li, Yuan Yao, and Hanghang Tong. 2023. Towards robust neural graph collaborative filtering via structure denoising and embedding perturbation. *ACM Transactions on Information Systems* 41, 3 (2023), 1–28.
[58] Yang Ye and Shihao Ji. 2021. Sparse graph attention networks. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2021), 905–916.
[59] Baolin Yi, Xiaoxuan Shen, Hai Liu, Zhaoli Zhang, Wei Zhang, Sannyuya Liu, and Naixue Xiong. 2019. Deep matrix factorization with implicit feedback embedding for recommendation system. *IEEE Transactions on Industrial Informatics* 15, 8 (2019), 4591–4601.
[60] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *NeurIPS* 33 (2020), 5812–5823.
[61] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? Simple graph contrastive learning for recommendation. In *SIGIR*, 1294–1303.
[62] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous graph neural network. In *KDD*, 793–803.
[63] Mengmei Zhang, Xiao Wang, Meiqi Zhu, Chuan Shi, Zhiqiang Zhang, and Jun Zhou. 2022. Robust heterogeneous graph neural networks against adversarial attacks. In *AAAI*, Vol. 36, 4363–4370.
[64] Peitian Zhang, Zheng Liu, Shitao Xiao, Zhicheng Dou, and Jing Yao. 2023. Hybrid inverted Index is a robust accelerator for dense retrieval. In *EMNLP*, 1877–1888.
[65] Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023. Retrieve anything to augment large language models. arXiv:2310.07554. Retrieved from https://arxiv.org/abs/2310.07554
[66] Yi Zhang, Yiwen Zhang, Lei Sang, and Victor S. Sheng. 2024. Simplify to the limit! Embedding-less graph collaborative filtering for recommender systems. *ACM Transactions on Information Systems* 43, 1 (2024), 1–30.
[67] Yi Zhang, Yiwen Zhang, Dengcheng Yan, Shuiguang Deng, and Yun Yang. 2023. Revisiting graph-based recommender systems from the perspective of variational auto-encoder. *ACM Transactions on Information Systems* 41, 3 (2023), 1–28.
[68] Chen Zhao, Le Wu, Pengyang Shao, Kun Zhang, Richang Hong, and Meng Wang. 2023. Fair representation learning for recommendation: A mutual information perspective. In *AAAI*, Vol. 37, 4911–4919.
[69] Jiawei Zheng, Qianli Ma, Hao Gu, and Zhenjing Zheng. 2021. Multi-view denoising graph auto-encoders on heterogeneous information networks for cold-start recommendation. In *KDD*, 2338–2348.

Received 26 January 2024; revised 30 August 2024; accepted 26 November 2024