RESEARCH-ARTICLE

Heterogeneous Adaptive Preference Learning for Recommendation

LEI SANG, Anhui University, Hefei, Anhui, China

WEICHEN FEI, Anhui University, Hefei, Anhui, China

YI ZHANG, Anhui University, Hefei, Anhui, China

YUEE HUANG, Wannan Medical College, Wuhu, Anhui, China

YIWEN ZHANG, Anhui University, Hefei, Anhui, China

Published: 29 July 2025
Online AM: 06 April 2024
Accepted: 04 March 2024
Revised: 04 March 2024
Received: 29 August 2023

Heterogeneous Adaptive Preference Learning for Recommendation

LEI SANG, Anhui University, Hefei, China
WEICHEN FEI, Anhui University, Hefei, China
YI ZHANG, Anhui University, Hefei, China
YUEE HUANG, Wannan Medical College, Wuhu, China
YIWEN ZHANG, Anhui University, Hefei, China

Graph-based collaborative filtering techniques have emerged as a promising recommendation approach by modeling user-item interaction as graphs. Recently, contrastive learning has been employed in graph collaborative through data augmentation, which can effectively offer data efficiency and reduce labeling costs. Nonetheless, most existing contrastive learning approaches overlook the heterogeneous auxiliary information pertaining to users and items, such as user social relationships and item categories, which are crucial to alleviate the data sparsity issue. In this paper, we propose a novel contrastive learning method, referred to as Heterogeneous Adaptive Preference Learning for Recommendation (HAPLRec), which explicitly incorporates fine-grained preference information from both users and items. Specifically, we construct user relationship graphs and item relationship graphs based on specific meta-paths in a heterogeneous graph. Subsequently, we conduct data augmentation on these graphs individually to obtain auxiliary contrastive tasks. Moreover, we introduce an optimization algorithm that leverages the gradient similarity between the main task and the auxiliary tasks, dynamically adjusting the weight assigned to each task to expedite achieving superior performance within a shorter time frame. The effectiveness of the proposed model is demonstrated through extensive experiments conducted on three publicly available datasets.

CCS Concepts: • **Information systems → Collaborative and social computing systems and tools;**

Additional Key Words and Phrases: Heterogeneous information network, contrastive learning, graph neural network, multi-task learning

ACM Reference Format:
Lei Sang, Weichen Fei, Yi Zhang, Yuee Huang, and Yiwen Zhang. 2026. Heterogeneous Adaptive Preference Learning for Recommendation. *ACM Trans. Recomm. Syst.* 4, 1, Article 3 (July 2026), 25 pages. https://doi.org/10.1145/3656480

The authors acknowledge the support of the National Science Foundation of China [No. 62272001 and No. 62206002], the Hefei Key Common Technology Project (GJ2022GX15), the Anhui Provincial Natural Science Foundation [No. 2208085QF195], the University Collaborative Innovation Project of Anhui Province [No. GXXT-2021-087] and the Anhui Province Key Research and Development Program [No. 202104a05020058].
Authors' addresses: L. Sang, W. Fei, and Y. Zhang, Anhui University, Hefei, Anhui, China; e-mails: sanglei@ahu.edu.cn, fei@stu.ahu.edu.cn, zhangyi.ahu@gmail.com; Y. Huang, Wannan Medical College, Wuhu, Anhui, China; e-mail: huangyewindow@163.com; Y. Zhang (Corresponding author), Anhui University, Hefei, China; e-mail: zhangyiwen@ahu.edu.cn.

ACM 2770-6699/2026/07-ART3
https://doi.org/10.1145/3656480

1 INTRODUCTION

Recommender systems have emerged as vital components in online platforms, such as e-commerce and news portals, aimed at mitigating the burden of information overload for users [49]. An efficient recommender system aims to accurately capture users' preferences and provide recommendations for items that align with their potential interests, thereby enhancing user satisfaction and retention. The success of a commercial platform is heavily dependent on the quality of its recommender system. **Collaborative filtering (CF)** [36, 64] has garnered substantial attention in both academic and industrial circles as a critical task within recommender systems. A prevailing approach in CF entails learning vector representations, referred to as embeddings, for users and items by leveraging historical interaction data. These embeddings are subsequently utilized to facilitate top-k recommendation by calculating pairwise similarity between user and item embeddings. The research community has shown great interest in this methodology due to its efficacy in capturing the latent factors influencing user-item interactions. **Matrix factorization (MF)** [16, 21] is a classic collaborative filtering technique that represents users and items using latent feature vectors in a shared latent space. However, its performance is limited by the simplistic inner product interaction function. To overcome this limitation, **neural collaborative filtering (NCF)** and **deep factorization machine (DeepFM)** were proposed, incorporating **multilayer perceptrons (MLP)** to improve interaction modeling [10, 15]. Despite these advancements, these methods still overlook the high-order structural information present in the observed data.

In recent times, **Graph Neural Networks (GNNs)** have achieved remarkable accomplishments in recommender systems owing to their exceptional capability in modeling graph-structured data. GNNs possess the ability to capture intricate, high-order interactions prevalent in user-item relationships by means of iterative propagation. The introduction of Pinsage [56] marks a significant milestone as it combines efficient random walks and graph convolutions to generate item embeddings that encapsulate both graph structure and item feature information. This represents the first instance where **Graph Convolutional Networks (GCN)** have been successfully applied to recommender systems operating at a web-scale. Another noteworthy development in this domain is NGCF [43], which introduces a novel graph-based collaborative filtering framework. NGCF utilizes a graph convolutional encoder to aggregate information from neighboring nodes within a user-item interaction graph, thereby acquiring the knowledge required to learn user and item embeddings. Although NGCF has showcased promising results, its design is characterized by a heavy and cumbersome structure, as it directly inherits numerous operations from GCN. Consequently, LightGCN [14] was introduced to alleviate these concerns. LightGCN simplifies the architecture by removing feature transformation and non-linear activation components, thereby facilitating easier training while simultaneously enhancing performance. However, most GNN-based collaborative recommender systems rely on rich records of user-item interactions, and the historical interaction data of users in the real world are often sparse.

In order to address the aforementioned issue, some researchers have introduced **self-supervised learning (SSL)** techniques into recommender systems [24, 48]. SSL has emerged as a learning paradigm that reduces the reliance on manual labels and enables training on vast amounts of unlabeled data [22, 31]. The core principle of SSL revolves around learning from automatically generated supervisory signals derived from raw data, offering a solution to the issue of data sparsity in recommender systems. SSL demonstrates immense potential for enhancing the quality of recommendations. One of the prominent methods in SSL is contrastive learning, which allows training models without explicit labels. This approach achieves this by learning discriminative embeddings from unlabeled sample data, wherein the distance between negative samples is maximized while the distance between positive samples is minimized. By incorporating the contrastive learning task as an auxiliary task within the recommendation framework, the recommendation

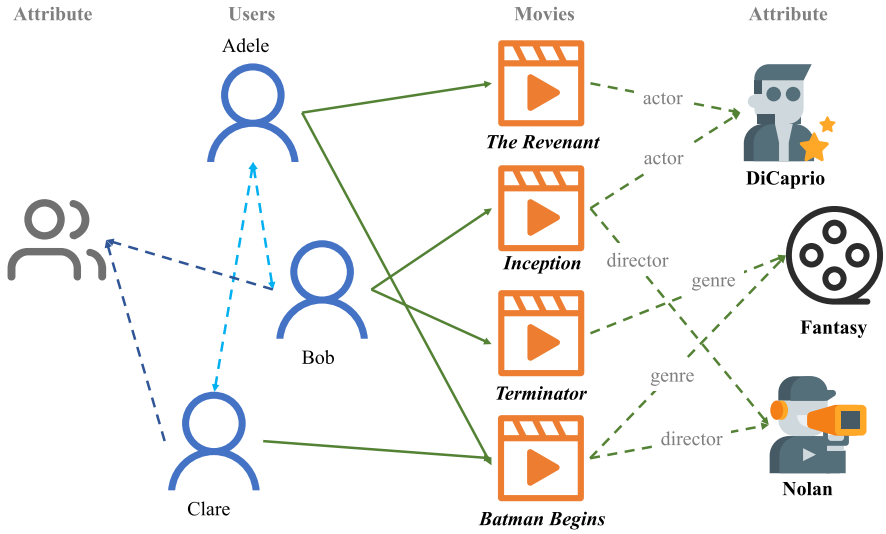


Fig. 1. The illustration for heterogeneous information network based recommendation scenario.

performance is further enhanced. Despite the ability of SSL to alleviate the data sparsity problem, current self-supervised recommender systems still encounter the following challenges:

— **Ignoring heterogeneous auxiliary data in GNN-based recommender systems.** Existing GNN-based contrastive learning recommender systems primarily leverage the user-item interaction graph for information extraction, while overlooking the auxiliary data linked to users and items, such as user social relationships and item characteristics. However, it is intuitively apparent that integrating these auxiliary sources can notably augment the recommender system's capacity to uncover more comprehensive insights. For example, a relationship between two movies can be inferred from: (1) "Movie-User-User-Movie" and (2) "Movie-Genre-Movie". These meta-paths capture the following semantic relations: (1) watched by friends; (2) belonging to the same genre. As shown in Figure 1, for example, the movie *The Revenant* and *Inception* can be connected through the path *The Revenant - Adele - Bob - Inception*, and can also be connected through the path *The Revenant - DiCaprio - Inception*.

— **Fixed coefficients in self-supervised recommender systems.** Most of the current self-supervised recommender systems regard the self-supervised task as an auxiliary task of the recommendation task. Most of the work is solved by tuning parameters, which will undoubtedly waste a lot of time and energy. In addition, setting fixed coefficients throughout the training process cannot achieve optimal results. Because different auxiliary tasks have different influences on the main task, the influence of auxiliary tasks is also constantly changing during the training process. So how to set an appropriate coefficient for the auxiliary task is a problem worth solving.

In this paper, we propose a novel contrastive learning method, referred to as **Heterogeneous Adaptive Preference Learning for Recommendation (HAPLRec)**, which explicitly incorporates side information with adaptive weights. (1) Firstly, we leverage heterogeneous auxiliary information available by a heterogeneous graph. Accordingly, to capture fine-grain semantic preference information, we conduct graph augmentation guided by various meta-paths. Each meta-path represents a distinct type of semantic preference information and serves as a

contrastive learning auxiliary task. (2) Furthermore, in order to address the issue of coefficients in multi-preference training, we propose the utilization of **adaptive preference learning (APL)**, an optimization algorithm designed specifically for multi-term preference loss functions. APL expands upon the existing Adam algorithm and enables the efficient training of unweighted multi-term preference objectives. Consequently, APL facilitates the computational-intensive task of hyper-parameter exploration, which is necessary for appropriately weighting the multiple preference. During each iteration of the training process, a dynamic weight is allocated to each individual preference loss term based on the magnitude of its gradient. This weight allocation scheme aims to strike a balance between the gradients and ensure their magnitudes are comparable.

Our HAPLRec is a model-agnostic approach that can be applied to any graph-based model encompassing user and/or item embeddings. In this work, we implement it on the simple yet effective model, LightGCN. Through experimental studies on three benchmark datasets, we demonstrate the effectiveness of HAPLRec. It not only significantly enhances the accuracy of the recommendations, particularly for long-tail items, but also substantially reduces the time required for hyperparameter tuning of the model. The contributions of this work can be summarized as follows:

— We propose data augmentation on meta-path-guided homogeneous graphs to make full use of heterogeneous information. By exploiting different meta-paths to capture different semantic information, it provides sufficient supervision signals for the main task of recommendation.
— We introduce the optimizer APL, which can dynamically balance multiple loss terms without manually setting coefficients for each auxiliary task.
— We conduct extensive experiments on three benchmark datasets to demonstrate the superiority of HAPLRec.

2 RELATED WORK

This section provides a comprehensive review of three tasks that are closely related to our research: graph-based recommendation, contrastive learning in recommendation, and multi-task learning.

2.1 Graph-based Recommendation

Collaborative Filtering (CF) is a widely used technique in modern recommender systems, known for its effectiveness [36, 64]. The underlying principle is that users with similar preferences tend to have similar tastes in items. However, CF can face challenges due to data sparsity and scalability. To address these issues, Matrix Factorization (MF) techniques have been introduced [16, 21]. MF decomposes the original sparse user-item matrix into low-dimensional matrices containing latent factors or features, thereby reducing sparsity. In recent years, deep learning-based models have been proposed to enhance recommendation systems [66]. These models leverage the power of deep learning to learn informative and compressed latent features. By combining neural networks and other advanced technologies (such as reinforcement learning [2, 42], variational auto-encoder [68], etc.), deep learning models can capture complex patterns and relationships in user-item interactions, thereby improving recommendation performance.

Graph Convolutional Networks (GCNs), extending the concept of Convolutional Neural Networks (CNNs) to graph-structured data, have emerged as potent tools for processing and analyzing data represented in graph form. In recent years, GCN-based collaborative filtering (CF) has received great attention and been extensively studied [49]. A user-item interaction matrix can be naturally viewed as a bipartite graph, where users and items are connected by edges representing their interactions. GCN-based CF leverages the inherent graph structure and integrates high-order information to improve recommendation performance. By leveraging GCN, the model can capture

complex relationships and dependencies between users and items, thereby improving recommendation quality. GC-MC [1] treats matrix completion in the recommendation task as a link prediction problem on a graph, and pioneers the application of graph neural networks to this task. The method utilizes graph information to improve the accuracy of predicting missing values. PinSage [56] introduces a groundbreaking method for building web-scale recommender systems by combining efficient random walks with graph convolutions. This approach generates node embeddings that reflect both the graph structure and node features. Additionally, to capture the cooperative signals between users and items, NGCF [43] explicitly incorporates collaborative filtering signals into the embedding process for the first time. LightGCN [14] simplifies the unnecessary nonlinear activation and feature transformation modules of NGCF, achieving better performance at a faster speed.

Moreover, the adoption of **heterogeneous information networks (HINs)** in recommendation systems has garnered increasing interest from both academic and industrial researchers. HeteRec [63] proposes a feature representation approach based on meta-paths to capture the relationships between users and items along diverse paths in related information networks. Hete-CF [27] proposes a social recommendation model based on collaborative filtering that incorporates heterogeneous relations. MCRec [17] leverages context derived from meta-paths in top-N recommendation. HERec [38] designs a meta-path-based random walk strategy to generate node sequences and employs deepwalk to generate node embeddings.

2.2 Contrastive Learning in Recommendation

Contrastive learning (CL) has become an influential method across various fields due to its ability to exploit large amounts of unlabeled data [4, 8, 57]. Particularly in recommender systems, where CL's self-supervised nature offers a targeted resolution to data sparsity problems [58, 61], its integration has seen rapid adoption [47, 60, 70]. The cornerstone of contemporary contrastive recommendation approaches lies in classifying each instance (such as users or items) as a distinct class. The aim is to pull identical instance views closer while pushing differing ones apart during representation learning. Views are created by applying various transformations to the original data, with the aim of maximizing the mutual information between views that come from the same instance. This process allows the recommendation model to better understand the core of user-item interactions.

S^3-Rec [71] represents a pioneering effort to meld CL with sequential recommendation. This method initially conceals selected attributes and items to form sequence augmentations, subsequently pre-training the Transformer [40] to foster consistency across different augmentations. Parallel developments, such as CL4SRec [54], have introduced further augmentation techniques, including item reordering and cropping. S^2-DHCN [51] and NCL [24] employ other strategies where advanced augmentations such as re-organizing/clustering sequential data for more potent self-supervised signals. DuoRec [33] incorporates model-level augmentation via encoder dropout, and Xia et al. [52] leverage CL in a self-supervised knowledge distillation framework to enhance next-item recommendations for models on resource-constrained devices.

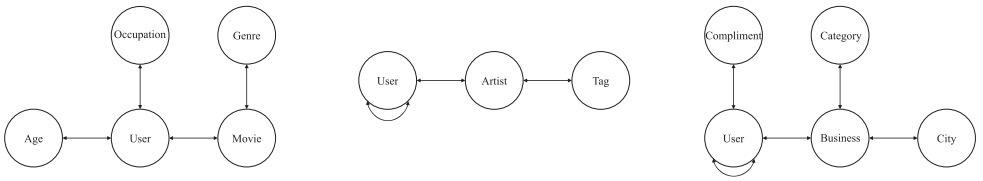
Concurrently, CL's integration has also expanded to various graph-based recommendation scenarios. For social recommendation, models like S^2-MHCN [60] and SMIN [26] are integrated with CL. HHGR [65] introduces a double-scale augmentation technique for group recommendation, providing a more refined contrastive objective for users and groups. In the realm of cross-domain and bundle recommendation, approaches such as CCDR [53] and CrossCBR [29] explore the applicability of CL. Yao et al. [55] develop a feature dropout-based two-tower architecture for large-scale item recommendation, while NCL [24] employs a prototypical contrastive objective to encapsulate the correlations between users or items and their context. Additional innovations include SEPT

[59] and COTREC [50], which leverage semi-supervised learning on perturbed graphs to unearth multiple positive samples for social or session-based recommendation. The most prevalently utilized model, SGL [48], implements edge or node dropout to enrich the graph data. HeCo [45] encodes nodes from the network schema view and meta-path view, and redefines the positive sample nodes of HIN. HGCL [3] incorporates heterogeneous relational semantics into user-item interaction modeling and enhances knowledge transfer from different viewpoints through contrastive learning.

2.3 Multi-task Learning

Multi-task learning (MTL) is a specialized training paradigm wherein a machine learning model is concurrently trained across multiple tasks, capitalizing on shared representations to extract commonalities across a cluster of related tasks [7, 35]. This communal structure enhances data efficiency and potentially accelerates learning for akin or subsequent tasks, thus addressing the prevalent deep learning challenges of intensive data prerequisites and computational overhead. Consequently, MTL has found extensive application across diverse domains, such as vision and language processing [13, 67]. In recent years, with the introduction of **deep reinforcement learning (DRL)**, multi-task learning has gradually attracted attention in the field of reinforcement learning [41]. A crucial component of MTL is the strategic manipulation of gradients to maintain equilibrium across the learning process of each task. In its most rudimentary form, gradient manipulation can be achieved by re-weighting individual task losses, utilizing criteria such as uncertainty [19], gradient norm [5], or difficulty [11]. While functional, these methods predominantly rely on heuristics and may suffer from inconsistencies in performance. Advanced gradient manipulations in MTL are conceptualized in various ways. [37] treats MTL as a multi-objective optimization problem, applying multiple gradient descent algorithms for precise optimization. PCGrad [62] identifies and addresses a significant optimization hurdle within MTL – conflicting gradients. It navigates this challenge by projecting each task gradient onto the orthogonal plane of other task gradients, subsequently merging them to compose the ultimate update vector. Though these methods deliver robust empirical results, they can only ascertain convergence to a Pareto-stationary point, leaving the exact convergence destination ambiguous. Further inventive gradient manipulations include GradDrop [6], a method that selectively excludes task gradients based on the degree of conflict, and IMTLG [25], which formulates an update vector with equalized projections on each task gradient. RotoGrad [18], a more nuanced approach, isolates the scaling and rotation of task gradients to alleviate optimization discord. These methodological advancements underscore the complexity of balancing multi-task learning and the ongoing pursuit of optimization techniques that are both robust and efficient.

In various personalized recommendation scenarios, incorporating auxiliary tasks through joint learning can improve the test accuracy of the target task [9, 28, 46]. This paradigm has also been adopted in recent self-supervised recommendation models, where the primary task involves the pairwise Bayesian Personalized Ranking loss, while the self-supervised learning task serves as an auxiliary task, and the two are jointly trained [24, 48, 59]. The success of these techniques hinges on the alignment between the auxiliary and target tasks, as well as their combination. In previous self-supervised recommendation models, the weights of each auxiliary task are often set manually based on prior intuition or through hyperparameter tuning. However, determining the weights can be challenging when there are many auxiliary tasks, and the usefulness of the auxiliary tasks to the primary task cannot be disregarded, as the degree of correlation between them may change during training. Although, as mentioned earlier, there are many related multi-task optimization techniques, no recommended techniques have been proposed. And there are



(a) Movielens (b) LastFM (c) Yelp

Fig. 2. Network schemas of heterogeneous information networks for three datasets.

primary and secondary multi-tasks in recommendation, so when dealing with gradient conflicts among them, it cannot be simply regarded as an ordinary multi-task optimization problem.

3 PRELIMINARY

3.1 Heterogeneous Information Network

Real-world graphs often consist of multiple types of nodes and edges, forming what is widely known as a heterogeneous graph. Modeling and learning on heterogeneous graphs require specialized considerations to effectively capture and represent the diverse nature of their nodes and edges. This complexity stems from the need to accommodate various types of relationships, attributes, and entities within the graph, making the task of building meaningful representations a challenging but vital aspect of working with heterogeneous structures.

Definition 1 (Heterogeneous Information Network (HIN) [39]). A HIN, represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is characterized by two mapping functions ϕ which maps nodes in $\mathcal{V}$ to their respective types in $\mathcal{A}$, and φ which maps edges in $\mathcal{E}$ to their associated types in $\mathcal{R}$. Here, $\mathcal{A}$ is the set of distinct node types and $\mathcal{R}$ is the set of distinct edge types. The condition $|\mathcal{A}| + |\mathcal{R}| > 2$ ensures the network's heterogeneity.

In a HIN, the concept of the **network schema** is utilized to depict the high-level topology of the network. Specifically, the network schema outlines the various types of nodes and the interaction relations between them. It serves as a structural blueprint that illustrates how different entities within the network interact and connect with each other, encompassing the diversity and complexity of the relationships in a HIN.

Definition 2 (Network Schema). The network schema is denoted as $\mathcal{S} = (\mathcal{A}, \mathcal{R})$. It is a meta-template for HIN as described in Definition 1, which comprehensively describes the structural patterns in HIN, and guides our mining of network semantics.

Example 1. Taking Figure 2(a) as an example, we show the network schema corresponding to the Movielens dataset, which contains various types of objects (e.g., User (U), Movie (M), Age (A), Occupation (O) and Genre (G)) and their semantic relations (e.g., 'watching' relation among users and movies, 'attribute' relation among users and ages/occupations, and 'attribute' relation among movies and genres).

In HIN, nodes can be connected via multiple types of semantic paths, which are defined as **meta-paths**.

Definition 3 (Meta-path). In a HIN, a meta-path ρ is a sequence of object types connected by link types. It is denoted as $\mathcal{A}_1 \xrightarrow{\mathcal{R}_1} \mathcal{A}_2 \xrightarrow{\mathcal{R}_2} \cdots \xrightarrow{\mathcal{R}_l} \mathcal{A}_{l+1}$ (abbreviated as $\mathcal{A}_1\mathcal{A}_2 \cdots \mathcal{A}_{l+1}$), where $\mathcal{A}_i$ is the object type and $\mathcal{R}_i$ is the relation type. A meta-path ρ defines a composite relation

Table 1. Notations and Description

Notations	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	a heterogeneous information network (HIN)
$\mathcal{V}$	HIN node set
$\mathcal{E}$	HIN link set
$\mathcal{A}$	HIN node type set
$\mathcal{R}$	HIN link type set
p	a meta-path
$\mathcal{U} = \mathrm{u}_1, \mathrm{u}_2, \ldots, \mathrm{u}_{\lvert\mathcal{U}\rvert}$	user set
$\mathcal{I} = \mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_{\lvert\mathcal{I}\rvert}$	item set
$\mathbf{E}$	node representation
O^+	observed interaction set
O^-	unobserved interaction set
$\mathbf{e}_{\mathrm{u}_i}$	representation of user u_i
$\mathbf{e}_{\mathrm{i}_j}$	representation of user i_j
$\mathbf{G}$	gradient

$\mathcal{R}_1 \circ \mathcal{R}_2 \circ \cdots \mathcal{R}_l$ between types $\mathcal{A}_1$ and $\mathcal{A}_{l+1}$, capturing the semantics of connections in the HIN, where $\circ$ is a composition operator on relations.

Example 2. Take Figure 2(a) as an example. We can connect two movies with different meta-paths, such as "$Movie - Genre - Movie$" (MGM) and "$Movie - User - Movie$" (MUM). Commonly, different meta-paths can reveal the different inter-dependency information of two movies. The path "MGM" illustrates that two movies are of the same genre, while the path "MUM" indicates that two movies were watched by the same user.

In the recommendation scenario, different meta-paths can reveal users' distinct preferences for certain items. For instance, certain users may exhibit a preference for novel and unique products, which might be influenced by their inclination to seek advice and recommendations from friends or social circles before making a purchase. On the other hand, there are individuals who tend to favor familiarity and are more likely to buy products from the same brand they have purchased before. Such preferences are effectively captured by the diverse meta-paths utilized in the recommendation system.

Notations used throughout the article can be found in Table 1.

3.2 GCN for Recommendation

In this subsection, we outline the typical structure of GCN-based collaborative filtering models. Let $\mathcal{U}$ be the set of users and $\mathcal{I}$ be the set of items. The observed interactions can be denoted as $O^+ = \{y_{ij} | \mathrm{u}_i \in \mathcal{U}, \mathrm{i}_j \in \mathcal{I}\}$, where y_{ij} signifies that user u_i has previously interacted with item i_j. In general, most existing models create a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents a node set and $\mathcal{E}$ represents an edge set.

Graph Convolutional Networks (GCN) are designed to process data structured as graphs. The core idea is to update the representation of a node by aggregating features from its neighboring nodes in the graph. Formally, given a graph $\mathcal{G}$, the iterative update at the l-th layer is defined as:

$$\mathbf{E}^{(l)} = H(\mathbf{E}^{(l-1)}, \mathcal{G}), \tag{1}$$

where $\mathbf{E}^{(l)}$ denotes node representations at the l-th layer, initialized as $\mathbf{E}^{(0)}$ for ID embeddings. The function H signifies the neighborhood aggregation, which can be expanded as:

$$\mathbf{e}_{\mathrm{u}_i}^{(l)} = f_{\text{combine}}(\mathbf{e}_{\mathrm{u}_i}^{(l-1)}, f_{\text{aggregate}}(\{\mathbf{e}_{\mathrm{i}_j}^{(l-1)} | \mathrm{i}_j \in \mathcal{N}_{\mathrm{u}_i}\})). \tag{2}$$

Here, the representation of node u_i at layer l is aggregated from the features of its neighbors $\mathcal{N}_{\mathrm{u}_i}$ from the previous layer. The functions $f_{\text{aggregate}}(\cdot)$ and $f_{\text{combine}}(\cdot)$ are determined by the specific GCN design, providing flexibility in the aggregation mechanism.

After L layers of iteration, the final node representation can be obtained via a readout function:

$$\mathbf{e}_{\mathrm{u}_i} = f_{\text{readout}}\Big(\{\mathbf{e}_{\mathrm{u}_i}^{(l)} | l = [0, \ldots, L]\}\Big). \tag{3}$$

Common choices for the readout function include using only the last-layer representation, concatenation across layers, or a weighted sum of representations from different layers.

In this paper, the GCN we adopt is LightGCN [14], which is the state-of-the-art GCN-based method in CF. Its information propagation rule is (taking user u_i as an example):

$$\mathbf{e}_{\mathrm{u}_i}^{(l)} = \sum_{\mathrm{i}_j \in \mathcal{N}_{\mathrm{u}_i}} \frac{\mathbf{e}_{\mathrm{i}_j}^{(l-1)}}{\sqrt{|\mathcal{N}_{\mathrm{u}_i}|\,|\mathcal{N}_{\mathrm{i}_j}|}}. \tag{4}$$

After going through L layers, the final representation of the user u_i is calculated as:

$$\mathbf{e}_{\mathrm{u}_i} = \sum_{l=0}^{L} \alpha_l \mathbf{e}_{\mathrm{u}_i}^{(l)}, \tag{5}$$

where α is usually set to $1/L + 1$.

4 METHODOLOGY

In this section, we first give an overview of the proposed framework, and then detail the individual components of the model, including the construction method of the user/item relationship graph, the construction of contrastive learning auxiliary tasks, and the optimization strategy for multi-task learning.

4.1 An Overview of the Proposed Framework

The architecture of the proposed framework is shown in Figure 3. First, we construct a user relationship graph and an item relationship graph based on heterogeneous information, and a user-item graph based on users' historical interaction data. Then, we use a graph neural network to encode users and items; at the same time, we perform data augmentation based on the user relationship graph and item relationship graph, and then perform contrastive learning based on the generated representations to construct auxiliary tasks. Finally, we implement dynamic adjustments to multi-task learning using a gradient adaptive strategy.

4.2 Neighborhood Aggregation Based on Preference Guidance

To understand fine-grained preferences, we first extract information about user relationships and dependencies between items from a collaborative heterogeneous graph $\mathcal{G}$.

We extract various meta-path instances for both user and item domains by investigating the multifaceted dependence effects that are influenced by social and knowledge aspects. Here, we use p_j^u and p_j^i to denote the j-th meta-path instance extracted for users and items, respectively. The corresponding sets of these instances are represented as P^u and P^i such that $p_j^u \in P^u, p_j^i \in P^i$.

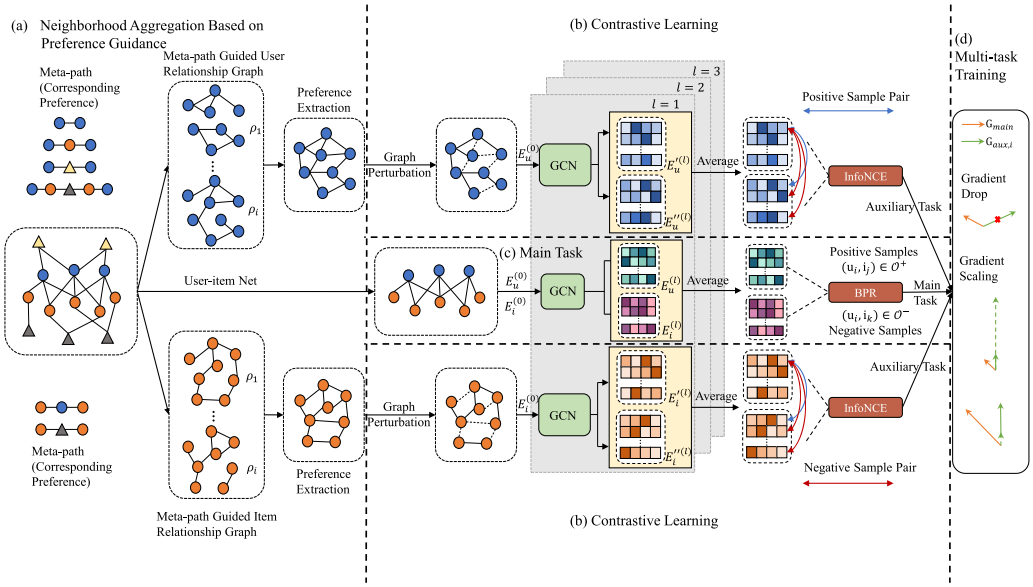


Fig. 3. Overall architecture of the HAPLRec model. (a) describes the process of using preference guidance to construct user relationship graphs and item relationship graphs; (b) contains three important components of contrastive learning: graph perturbation, encoder, and loss function; (c) is the classic recommended main task; (d) includes strategies for multi-task learning.

In the recommendation scenario, different meta-paths can reveal users' distinct preferences for certain items. For instance, certain users may exhibit a preference for novel and unique products, which might be influenced by their inclination to seek advice and recommendations from friends or social circles before making a purchase. On the other hand, there are individuals who tend to favor familiarity and are more likely to buy products from the same brand they have purchased before. Such preferences are effectively captured by the diverse meta-paths utilized in the recommendation system. By considering these user-specific inclinations and leveraging the information obtained from various meta-paths, the recommendation model can cater to the heterogeneous preferences of users, providing them with personalized item recommendations. As a result, users who are influenced by their friends' opinions will be presented with items that align with their preferences for novelty, while those who prefer brand loyalty will receive recommendations based on their previous purchases. This approach enhances the overall effectiveness and accuracy of the recommendation system, ultimately leading to improved user satisfaction and engagement. We design meta-path instances by the following assumptions:

Assumption 1 (Friend Influence). *The preferences of a user's friends can influence their own interests. For instance, if user* u_i *enjoys watching movie* m_k*, his friend* u_j *is likely to enjoy* m_k *as well, even if* u_j *did not like it initially. This is because* u_j *may watch movie* m_k *with* u_i *due to* u_i*'s invitation and end up liking it because* u_i *likes it. The user relationship instance derived from this assumption is denoted by* UU*.*

Assumption 2 (User Behavior Similarity). *Users who purchase the same items may have similar interests and preferences. For example, if user* u_i *and user* u_j *both watched a less popular movie* m_k*, their preferences may be similar. The user relationship instance obtained based on the user behavior similarity assumption is* UMU*.*

Assumption 3 (Attribute-Based Similarity). *Users who purchase items from the same attribute may have similar interests and preferences. For instance, if user* u_i *and user* u_j *both watched movies* m_k *and* m_l *from category* a_n*, their preferences may be similar. The user relationship instance derived from the category-based similarity assumption is UMAMU.*

Assumption 4 (User-Based Similarity). *Items purchased by the same user may share some similarity. For example, if user* u_i *likes both movie* m_k *and movie* m_l*, then* m_k *and* m_l *may have some similarity. The item relationship instance derived from the user-based similarity assumption is MUM.*

Assumption 5 (Attribute-Based Item Similarity). *Items with the same attributes share a certain degree of similarity. For example, if movie* m_k *and movie* m_l *have the same attribute, then they may have some similarity. The item relationship instance derived from the category-based item similarity assumption is MAM.*

We obtain the corresponding relationship graph for each assumption by using a simple matrix multiplication with the meta-path instance obtained from the assumption. However, the resulting user relationship graph or item relationship graph is typically much denser than the user-item graph. Incorporating these augmented relationship graphs directly into the optimization procedure can lead to the introduction of numerous unreliable or noisy user-user or item-item pairs, thereby posing challenges in effectively training the model. To solve this problem, we select for each user or item the top k users or items with the most similar preferences. For example, consider the user relationship graph $\mathcal{G}_{UMU}$, which we normalize to obtain a preference similarity graph, namely:

$$\mathcal{G}^{'}_{UMU} = D^{-\frac{1}{2}}\mathcal{G}_{UMU}D^{-\frac{1}{2}}, \tag{6}$$

where D is the diagonal node degree matrix of $\mathcal{G}_{UMU}$. Therefore, the element $\mathcal{G}^{'}_{UMUi,j}$ in the matrix $\mathcal{G}^{'}_{UMU}$ is $\frac{\mathcal{G}_{UMU}}{\sqrt{d_i}\sqrt{d_j}}$.

The measure $\mathcal{G}^{'}_{UMUi,j}$ intuitively represents the preference similarity between user u_i and user u_j, as it is directly proportional to the number of co-occurrences of the two users and inversely proportional to their degree. Based on this measure, we select the top k most similar users, which enables us to reconstruct a new user relationship graph $\mathcal{G}^{k}_{UMU}$ with relatively less noise, using the meta-path instance UMU. Similarly, we can obtain user relationship graphs and item relationship graphs reconstructed from other meta-path instances. To incorporate multiple types of preference similarity, we merge several user relationship graphs with different meta-paths. For simplicity, we take the union of the relationship graphs obtained from multiple meta-path instances as the final relationship graph. This can be expressed as:

$$\begin{aligned}\mathcal{G}^{k}_{U} &= \mathcal{G}^{k}_{UU} \cup \mathcal{G}^{k}_{UMU} \cup \mathcal{G}^{k}_{UMAMU},\\ \mathcal{G}^{k}_{M} &= \mathcal{G}^{k}_{MUM} \cup \mathcal{G}^{k}_{MAM}.\end{aligned} \tag{7}$$

A more general formula is expressed as:

$$\mathcal{G}^{k}_{V} = \mathcal{G}^{k}_{p^{v}_{1}} \cup \mathcal{G}^{k}_{p^{v}_{2}} \cup \cdots \cup \mathcal{G}^{k}_{p^{v}_{|P^{v}|}}, \tag{8}$$

where $\mathcal{G}^{k}_{V}$ represents the final user/item relationship graph, p^{v}_{i} represents the i-th meta-path based on user/item ($i = 1, 2, \ldots, |P^{v}|$), and k represents the selection of the k most similar users/items under each meta-path.

4.3 Contrastive Learning

After obtaining the user relationship graph $\mathcal{G}_U^k$ and item relationship graph $\mathcal{G}_M^k$, we set up contrastive learning tasks for users and items, respectively. First, we perform graph perturbation on user relationship graph and item relationship graph. Following [48], we use three perturbation operators as follows:

— **Node Dropout (ND)**. With a given probability ρ, each node is removed from the graph, along with its connected edges.
— **Edge Dropout (ED)**. It drops out the edges in graph with a dropout ratio ρ.
— **Random Walk (RW)**. To enhance the model's capabilities, we apply various ED operations across different layers, leading to an outcome similar to Random Walk, as detailed in [32]. This approach deviates from the previous two operators, which generate a shared subgraph across all graph convolution layers.

After performing graph perturbation on the user/item relationship graph, we feed the generated subgraphs into the GCN encoder to obtain two representations of users/items respectively. In this work, as mentioned before, we adopt the LightGCN encoder, whose information propagation rules are adjusted according to Equation (4) (taking users as an example):

$$\mathbf{e}_{\mathrm{u}_i}^{(l)} = \sum_{\mathrm{u}_j \in \mathcal{N}_{\mathrm{u}_i}} \frac{\mathbf{e}_{\mathrm{u}_j}^{(l-1)}}{\sqrt{|\mathcal{N}_{\mathrm{u}_i}|\,|\mathcal{N}_{\mathrm{u}_j}|}}. \tag{9}$$

And then, we regard the same node on the user relationship graph as a positive sample pair (i.e., $\{(\mathbf{e}'_{\mathrm{u}_i}, \mathbf{e}''_{\mathrm{u}_i})|\mathrm{u}_i \in \mathcal{U}\}$), and the rest of the nodes as negative sample pairs (i.e., $\{(\mathbf{e}'_{\mathrm{u}_i}, \mathbf{e}''_{\mathrm{u}_j})|\mathrm{u}_i, \mathrm{u}_j \in \mathcal{U}, i \neq j\}$). Contrastive learning promotes the consistency between different views of the same node through positive sample pairs, while enhancing the discriminability between different nodes through negative sample pairs. Formally, following SimCLR [4], we adopt the contrastive loss, InfoNCE [12], to maximize the similarity of positive pairs and minimize the similarity of negative pairs. The mathematical formulation can be represented as:

$$\mathcal{L}_{aux}^{user} = \sum_{\mathrm{u}_i, \mathrm{u}_j \in \mathcal{U}} -\log \frac{\exp(s(\mathbf{e}'_{\mathrm{u}_i}, \mathbf{e}''_{\mathrm{u}_i})/\tau)}{\sum_{\mathrm{u}_j \in \mathcal{U}} \exp(s(\mathbf{e}'_{\mathrm{u}_i}, \mathbf{e}''_{\mathrm{u}_j})/\tau)}, \tag{10}$$

where $s(\cdot)$ stands for a function determining the similarity between two vectors, typically the cosine similarity, with τ as the temperature parameter for the softmax function. The item side's contrastive loss can also be derived, symbolized as $\mathcal{L}_{aux}^{item}$.

4.4 Multi-task Training

After several GCN layers, we can get the final representations of users and items, and then predict how likely user u_i is to like item i_j. We follow the previous method [14, 43] and use the most commonly used inner product to predict:

$$\hat{y}_{ij} = \mathbf{e}_{\mathrm{u}_i}^{\top} \mathbf{e}_{\mathrm{i}_j}. \tag{11}$$

To optimize model parameters, it has recently become a common approach to jointly train a self-supervised learning task as an auxiliary task and a recommendation main task. For the main recommendation task, the supervisory signal comes from the historical interaction records of users and items (i.e., the edges of $\mathcal{G}$). It is more common to use Paired **Bayesian Personalized Ranking (BPR)** loss [34]. Its basic idea is: given a user and two items, the model needs to rank the items that the user prefers before the items that the user dislikes less , so as to learn the user's

personalized preferences:

$$\mathcal{L}_{main} = \sum_{(u_i, i_j, i_k) \in O} -\log \sigma(\hat{y}_{ij} - \hat{y}_{ik}), \tag{12}$$

where $O = \{(u_i, i_j, i_k) | (u_i, i_j) \in O^+, (u_i, i_k) \in O^-\}$. The auxiliary tasks are contrastive learning tasks designed by each model, usually using InfoNCE loss (as shown in Equation (10)). Next, most methods adopt a multi-task training strategy to jointly optimize these tasks:

$$\mathcal{L}_{total} = \mathcal{L}_{main} + \sum_{i=1}^{K} \lambda_i \mathcal{L}_{aux,i} + \lambda_0 \|\Theta\|_2^2, \tag{13}$$

where Θ is the trainable parameter set , λ_0 is a hyperparameter to control the strengths of L_2 regularization. $\lambda_i (i = 1, \ldots, K)$ are used to control the strength of each auxiliary task. In this paper, $\mathcal{L}_{aux,i}$ corresponds to $\mathcal{L}_{aux}^{user}$ and $\mathcal{L}_{aux}^{item}$. Currently, the commonly used approach is to adjust the parameters λ_i through grid or random search. However, such an approach can be inefficient, consuming excessive time and resources. Moreover, when dealing with numerous auxiliary tasks, it might not always yield the optimal parameters. This inefficiency stems from the dynamic nature of the model during the training process; as the model evolves, so does the relative importance of each auxiliary task to the main task. Hence, using fixed weights in this scenario is clearly not the most rational choice. Therefore, inspired by [30], we have adopted a dynamic tuning strategy based on gradient magnitude, called **adaptive preference learning (APL)**. This method automatically balances the relationship between multiple auxiliary tasks and the main task, thereby adaptively learning various preferences.

The gradient $\mathbf{G}_{total}^t$ at the t-th iteration can be obtained from Equation (13):

$$\mathbf{G}_{total}^t = \nabla_\Theta \mathcal{L}_{total}^t = \nabla_\Theta \mathcal{L}_{main}^t + \sum_{i=1}^{K} \nabla_\Theta \lambda_i \mathcal{L}_{aux,i}^t = \mathbf{G}_{main}^t + \sum_{i=1}^{K} \lambda_i \mathbf{G}_{aux,i}^t, \tag{14}$$

where $\mathbf{G}_{total}^t$ is equivalent to the weighted summation of gradients from both the primary and auxiliary tasks. As Equation (14) indicates, if we ignore the parameters λ_i, the larger the gradient magnitude of the auxiliary task, the greater its impact on updating Θ. Therefore, we can control the impact of each task on parameters based on gradient size, which is the core idea of APL.

The implementation of APL consists of the following steps:

(1) Gradient calculation for main task. In each training iteration, we firstly calculate $\mathbf{G}_{main}^t$.
(2) Gradient initialization for auxiliary tasks. For each auxiliary task, we compute their gradients $\mathbf{G}_{aux,i}^t$ separately.
(3) Gradient rescaling for auxiliary tasks. Next, normalize each $\mathbf{G}_{aux,i}^t$ and then scale it to align with the magnitude of $\mathbf{G}_{main}^t$. Furthermore, to highlight the main task, we design a fixed coefficient λ to shrink the gradient size of the auxiliary tasks.
(4) Total gradient computation and parameter update. Construct $\mathbf{G}_{total}^t$ by summing $\mathbf{G}_{main}^t$ with the rescaled gradients from the auxiliary tasks. $\mathbf{G}_{total}^t$ is used to update the model parameter Θ by an optimizer such as Adam.

The APL procedure is iteratively executed until either convergence is achieved or the preset maximum iteration count is exceeded. Notably, the term $\frac{\|\mathbf{G}_{main}^t\|}{\|\mathbf{G}_{aux,i}^t\|}$ serves as a dynamic weighting mechanism for $\mathbf{G}_{aux,i}^t$. This dynamic adaptation promotes a balance between the main task and auxiliary gradients throughout the learning phase.

However, the disadvantages of this basic version are also obvious: it ignores the problem of gradient orientation. For example, when the gradient direction of the main task is opposite to that of

ALGORITHM 1: Adaptive Preference Learning.

Input: Θ^0: initial parameters, $\mathcal{L}_{main}$: main task , $\mathcal{L}_{aux,1}, \ldots, \mathcal{L}_{aux,K}$ auxiliary tasks, λ: coefficient of auxiliary tasks.
Output: Θ^T: final parameters.

while $t = \{1, \ldots, T\}$ **do**
  Calculate the gradient of the main task:
  $\mathbf{G}^t_{main} = \nabla_\Theta \mathcal{L}^t_{main}$
  for *i={1, …,K}* **do**
    Calculate initial gradients for auxiliary tasks:
    $\mathbf{G}^t_{aux,i} = \nabla_\Theta \mathcal{L}^t_{aux,i}$
    Calculate the gradient cosine similarity:
    $cosine = \frac{\mathbf{G}^t_{aux,i}\mathbf{G}^t_{main}}{\|\mathbf{G}^t_{aux,i}\|\|\mathbf{G}^t_{main}\|}$
    Rescale the gradient of the auxiliary task:
    if ($cosine <= 0$) **then**
     $\mathbf{G}^t_{aux,i} = 0$
    else
     $\mathbf{G}^t_{aux,i} = \lambda * \mathbf{G}^t_{aux,i} * \frac{\|\mathbf{G}^t_{main}\|}{\|\mathbf{G}^t_{aux,i}\|}$
  end
  Calculate the total gradient:
  $\mathbf{G}^t_{total} = \mathbf{G}^t_{main} + \mathbf{G}^t_{aux,1} + \cdots + \mathbf{G}^t_{aux,K}$
  Update Θ with $\mathbf{G}^t_{total}$
end

the auxiliary task, even if we control the gradient size of the two to the same magnitude, this will greatly inhibit the update of the main task and reduce the performance of the main task. To avoid this situation, we control the total gradient according to the gradient similarity between the main task and the auxiliary task. Its idea is: when the gradient similarity between the auxiliary task and the main task is less than or equal to 0, we cut the gradient of the auxiliary task. The key steps of our proposed approach are outlined in Algorithm 1. As an optimization algorithm, Adaptive Preference Learning is mainly designed to handle multiple loss terms in multi-task learning, especially when there is a main task and multiple auxiliary tasks. It has the potential to be applied to deep reinforcement learning scenarios where multiple objectives need to be optimized. The dynamic balancing mechanism of Adaptive Preference Learning, which allows for the continuous adjustment of gradient magnitudes for each loss term at different layers of the neural network, could potentially be beneficial in DRL setups where multiple objectives or reward functions need to be optimized simultaneously.

5 EXPERIMENTS

In this section, we demonstrate the effectiveness of **HAPLRec** by conducting experiments on three real-world datasets, comparing it to state-of-the-art recommendation methods.

Table 2. Statistics of the Three Datasets

Dataset (Density)	Relations (A-B)	Number of A	Number of B	Number of (A-B)
Movielens (6.30%)	User-Movie	943	1,682	100,000
	User-Age	943	8	943
	User-Occupation	943	21	943
	Movie-Genre	1,682	18	2,861
LastFM (0.28%)	User-Artist	1,892	17,632	92,834
	User-User	1,892	1,892	18,802
	Artist-Tag	17,632	11,945	184,941
Yelp (0.09%)	User-Business	16,239	14,284	198,397
	User-User	10,580	10,580	158,590
	User-Compliment	14,411	11	76,875
	Business-City	14,267	47	14,267
	Business-Category	14,180	511	40,009

5.1 Experimental Settings

Dataset. In this paper, we use three widely recognized datasets from distinct domains: the Movielens,[1] the LastFM,[2] and the Yelp[3] from movie, music and business domain, repectively. The Movielens dataset comprises 943 users and 1,682 movies, encompassing a total of 100,000 movie ratings on a scale of 1 to 5. In addition to the rating information, this dataset includes attribute data relevant to both users and movies. In contrast, the Yelp dataset focuses on local businesses and contains user ratings, social relationships, and business attributes. It consists of 16,239 users and 14,284 local establishments, with a total of 198,397 ratings ranging from 1 to 5. Moving to the domain of music, the LastFM dataset consists of 1,892 users and 17,632 artists, along with 92,834 listening records. These records can be seamlessly transformed into implicit feedback, and the dataset further includes social relations and attribute data pertaining to the artists. To handle the Movielens and Yelp datasets, we adopt the approach introduced in [17, 43], where a rating is considered as an interaction record, indicating whether a user has rated an item.

Table 2 presents a comprehensive overview of the characteristics of these three datasets. The initial row for each dataset provides information about the number of users, items, and interactions, while the subsequent rows present statistics related to other relationships within the data. It is important to note that these datasets exhibit variations in rating sparsity: the Yelp dataset demonstrates a high level of sparsity, whereas the Movielens dataset displays higher density. The specific meta-paths chosen for each dataset are outlined in Table 3.

Evaluation Protocol and Metrics. To evaluate recommendation performance, we randomly partition the entire user implicit feedback records of each dataset into training and test sets, utilizing 80% for training to predict the remaining 20%, as described in [17]. We also reserve 10% of the training data as a validation set for parameter tuning. The experiments in this section are conducted 5 times, with all items ranked, and the average result is reported, ensuring a rigorous and unbiased

[1] https://grouplens.org/datasets/movielens/
[2] https://www.last.fm
[3] http://www.yelp.com/dataset-challenge

Table 3. The Selected Meta-paths for Three Datasets in Our Work

Dataset	Meta-paths
Movielens	UMU, UAU, UOU, UMGMU, MUM, MGM
LastFM	UAU, UU, UATAU, AUA, ATA, AUUA
Yelp	UBU, UBCaBU, UU, BUB, BCiB, BCaB, BUUB

evaluation. For the top-N recommendation task, we follow the evaluation metrics commonly used in information retrieval, as in [17, 43], specifically employing Recall at rank K (Recall@K) and Normalized Discounted Cumulative Gain at rank K (NDCG@K), with K set to 20. The final results are first averaged across all test items for each user, and then across all users.

Baselines. In this study, we examine three prominent algorithms that represent different facets of collaborative filtering: classical collaborative filtering algorithms, including BPRMF and NeuMF; graph neural network-based collaborative filtering recommendation algorithms, encompassing GC-MC, NGCF, and LightGCN; and self-supervised recommendation algorithms, comprising Mult-VAE, SGL, and NCL. In addition, we also compared several recommendation methods based on heterogeneous information networks such as HERec, HAN, HeCo and HGCL. The comparison methods employed for evaluation are outlined below:

— **BPRMF** [34] is a bayesian personalized ranking matrix factorization algorithm that predicts user-item preferences by optimizing a loss function based on pairwise comparisons of items.
— **NeuMF** [15] uses a multi-layer perceptron to capture the nonlinear features of user-item interactions.
— **HERec** [38] is a HIN-based recommendation method that utilizes a meta-path-based random walk strategy to embed heterogeneous information networks, which are then integrated into an extended matrix factorization model.
— **HAN** [44] is a classic heterogeneous graph model. Its idea is that different types of edges should have different weights, and within the same type of edge, different neighbor nodes should have different weights. So it uses node-level attention and semantic-level attention.
— **HeCo** [45] proposes to learn node embeddings through HIN's network schema view and meta-path view to simultaneously capture local and high-order structures. On this basis, the concept of cross-view contrastive learning is proposed, and a view mask mechanism is proposed to extract positive and negative embeddings from two views.
— **HGCL** [3] proposes a customized contrastive learning framework that designs a meta-network to encode the personalized characteristics of users and items.
— **GC-MC** [1] adopts GCN [20] encoder to learn the embedding representations of both users and items by incorporating the information from the interaction graph.
— **NGCF** [43] represents the user-item interactions as a bipartite graph and uses GCNs to learn the embeddings of users and items by propagating the information through the graph.
— **LightGCN** [14], reported as the state-of-the-art GCN-based recommendation model, is a simplified version of NGCF. It removes redundant feature transformations and nonlinear activation functions.
— **Mult-VAE** [23] is a **variational autoencoder (VAE)** based CF method that introduces a generative model with multinomial likelihood.
— **SGL** [48] enhances the performance of recommendation tasks by designing a self-supervised auxiliary task. We adopt SGL-ED as the instantiation of SGL.
— **NCL** [24] considers both the comparison of user/item neighbors on the graph structure and its semantic neighbors by clustering node embeddings.

Table 4. Performance Comparison for Different Methods on Three Benchmarks

Method	Movielens		LastFM		Yelp	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
BPRMF	0.3442	0.4003	0.2474	0.2368	0.0480	0.0305
NeuMF	0.3288(−4.47%)	0.3815(−4.70%)	0.2300(−7.03%)	0.2148(−9.29%)	0.0430(−11.67%)	0.0250(−9.60%)
HERec	0.3495(+1.54%)	0.4122(+2.97%)	0.2493(+0.77%)	0.2383(+0.63%)	0.0766(+59.58%)	0.0604(+98.03%)
HAN	0.3249(−5.61%)	0.3820(−4.57%)	0.2478(+0.16%)	0.2286(−3.46%)	0.0792(+65.00%)	0.0494(+61.97%)
HeCo	0.3255(−5.43%)	0.3821(−4.55%)	0.2510(+1.46%)	0.2364(−0.17%)	0.0864(+80.00%)	0.0518(+69.84%)
HGCL	0.3510(+1.98%)	0.4145(+3.55%)	0.2572(+3.96%)	0.2449(+3.42%)	0.0871(+81.46%)	0.0539(+76.72%)
GC-MC	0.3489(+1.37%)	0.3912(+2.82%)	0.2432(−1.70%)	0.2277(−3.84%)	0.0638(+32.92%)	0.0371(+21.64%)
NGCF	0.3503(+1.77%)	0.4134(+3.27%)	0.2615(+5.70%)	0.2514(+6.17%)	0.0893(+86.04%)	0.0546(+79.02%)
LightGCN	0.3536(+2.73%)	0.4151(+3.70%)	0.2743(+10.87%)	0.2658(+12.55%)	0.0937(+95.21%)	0.0579(+89.84%)
Mult-VAE	0.3596(+4.47%)	0.4174(+4.27%)	0.2660(+7.52%)	0.2536(+7.09%)	0.0974(+102.92%)	0.0591(+93.77%)
SGL	0.3604(+4.71%)	0.4230(+5.67%)	0.2813(+13.70%)	0.2743(+15.84%)	0.0997(+107.71%)	0.0618(+102.62%)
NCL	0.3609(+4.85%)	0.4171(+4.20%)	0.2822(+14.07%)	0.2747(+16.01%)	0.1063(+121.46%)	0.0659(+116.07%)
HAPLRec-ND	0.3664(+6.45%)	0.4273(+6.74%)	0.2836(+14.63%)	0.2772(+17.06%)	0.1129(+135.21%)	0.0688(+125.57%)
HAPLRec-ED	0.3679(+6.89%)	0.4296(+7.32%)	**0.2864 (+15.76%)**	**0.2791(+17.86%)**	**0.1142(+137.92%)**	**0.0690(+126.23%)**
HAPLRec-RW	**0.3680(+6.91%)**	**0.4312(+7.72%)**	0.2840(+14.79%)	0.2770(+19.93%)	0.1138(+137.08%)	0.0687(+125.25%)

The percentage in parentheses indicates the performance improvement relative to BPRMF. Underlined indicates the best baseline, bold indicates the best result.

Implementation Details. All the baselines are implemented using RecBole,[4] a unified open-source framework for the development and reproduction of recommendation algorithms [69]. For a fair comparison, we employ the Adam optimizer to optimize all methods, and meticulously search the hyper-parameters across all baselines. The batch size is configured to 2,048, and all parameters are initialized following the default Xavier distribution. We fix the embedding size at 64 and apply early stopping with a patience of 50 epochs to mitigate overfitting, using Recall@20 as the guiding indicator. Our implementations are available at https://github.com/Feifei84/HAPLRec/tree/master.

5.2 Overall Performance Comparison

Table 4 presents a comprehensive performance comparison between our proposed model, **HAPLRec**, and various baseline methods across three distinct datasets. Analyzing the table yields several noteworthy observations:

(1) In the two classical collaborative filtering recommendation algorithms BPRMF and NeuMF, we found that the results of NeuMF did not meet expectations, probably because we did not pre-train as mentioned in [15].
(2) Basically all GNN-based recommendation models outperform traditional recommendation algorithms, thanks to the superiority of GNNs in modeling user-item interactions. Among them, the GC-MC method has the worst performance, because it only uses one layer of GCN,

[4] https://github.com/RUCAIBox/RecBole

ignoring the high-order information of users and items. The performance of the LightGCN method is better than that of GC-MC and NGCF, which is also consistent with the original report [14, 43].

(3) In the results of several models for heterogeneous recommendation, it can be seen that they perform better on sparse data sets, but perform relatively poorly on dense data sets. This is because the denser the data set, the more noise there is. The noise diffuses in the heterogeneous recommendation model and has a greater impact. The performance of the heterogeneous recommendation models HeCo and HGCL that introduced contrastive learning was further improved, which proved the effectiveness of contrastive learning.

(4) In the realm of self-supervised learning methods, their superiority over supervised learning approaches has been observed in numerous instances, signifying the profound benefits of incorporating self-supervised learning paradigms. However, when evaluating the performance on the Movielens dataset, it was noticed that Mult-VAE only exhibits a marginal improvement compared to LightGCN. We postulate that this discrepancy can be attributed to the fact that although Mult-VAE integrates a reconstruction loss as a self-supervised learning task, it lacks the utilization of supervisory signals in the form of a joint training main task. On the other hand, both SGL and NCL models are founded on recommendation tasks while being reinforced by self-supervised tasks, a design that has culminated in their notably commendable performance levels.

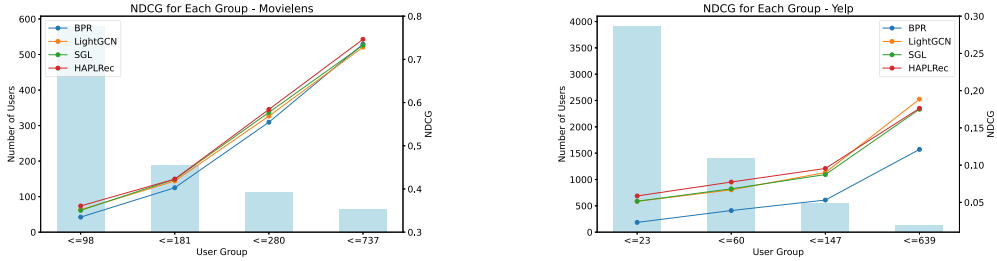
(5) The results obtained from our experimentation reveal that the three proposed models consistently exhibit superior performance when compared to the other baseline methods. Notably, HAPLRec-ED emerges as the most effective, displaying the highest overall performance. Following closely in second place is HAPLRec-RW demonstrating commendable performance as well. However, it is worth noting that HAPLRec-ND exhibits a comparatively weaker impact, possibly attributed to its excessive perturbation of the graph structure during the training process. This disruption leads to a more pronounced interference in the dissemination of information, resulting in training instability. On the contrary, HAPLRec-ED showcases the ability to adeptly capture inherent patterns within the graph structure, thereby contributing to notable performance improvements. Note that the HAPLRec in the following experiments all correspond to HAPLRec-ED.

5.3 Impact of Data Sparsity Levels

The problem of sparsity often hampers the effectiveness of recommender systems, as limited interactions from inactive users are inadequate for generating high-quality representations. In this context, we explore the potential benefits of exploiting contrastive learning and heterogeneous information to alleviate this limitation.

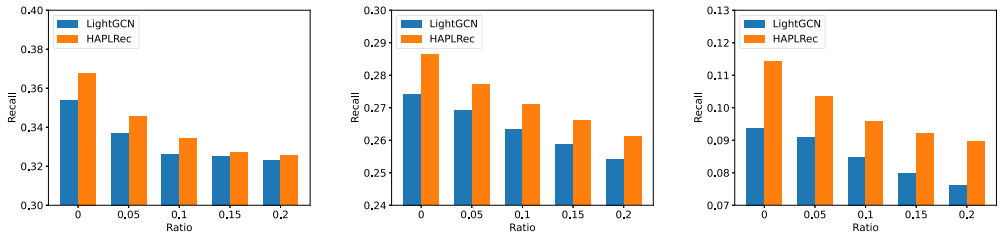
To systematically examine this aspect, we conduct experiments across user groups that are categorized by varying levels of sparsity. Specifically, we partition the test set into four distinct groups based on the number of interactions per user, ensuring that each group contains an equal sum of interactions. Illustratively, for the Movielens dataset, the thresholds for interaction numbers per user are set at 98, 181, 280, and 737, respectively. Figure 4 provides a visual representation of the outcomes with respect to NDCG@20 for different user groups in the Movielens and Yelp datasets. From this analysis, we discern that:

— HAPLRec and SGL outperform other baseline methods in all user groups, suggesting that employing self-supervised learning in recommendation tasks can effectively alleviate the sparsity problem of recommender systems by autonomously extracting supervisory signals from data.



(a) Movielens (b) Yelp

Fig. 4. Performance comparison of user groups with different sparsity levels on two datasets. The histograms represent the number of users in each group, and the lines show the NDCG@20 for each model under the corresponding group.



(a) Movielens (b) LastFM (c) Yelp

Fig. 5. Model performance w.r.t. noise ratio.

— Upon combined analysis of Figure 4(a) and 4(b), it is noticeable that there is an improvement in the first two groups (e.g., Movielens <= 98 and Yelp <= 23 groups improved by 2.91% and 13.78% from the best baseline respectively) than the other group improved significantly. Although the performance of our model is poor in the fourth group of data in the Yelp dataset, the overall performance is better than other models, which further verifies that our model is more beneficial to relatively inactive users.

5.4 Robustness to Noisy Interactions

We further examine the model's robustness to noisy interactions by conducting experiments with contaminated training sets. Specifically, we introduce adversarial examples by adding 5%, 10%, 15%, and 20% negative user-item interactions to the training set, while maintaining the testing set unchanged. The results of this experiment on the three datasets are shown in Figure 5.

The experimental results in Figure 5 show that the performance of both LightGCN and HAPLRec decreases as the noise ratio increases on the three datasets. But HAPLRec outperforms LightGCN on all three datasets, especially on the Yelp dataset. However, we found that as the noise ratio increases, the performance degradation ratio of HAPLRec is always higher than that of LightGCN. We think this is reasonable: as the proportion of negative interactions increases, more false edges will appear in the user relationship graph and item relationship graph constructed according to the heterogeneous graph, which will interfere with the training of the model. Therefore, HAPLRec is more affected by adding noise, resulting in a higher proportion of its performance drop. However, LightGCN only considers the information in the isomorphic graph. Relatively speaking, it is

Table 5. The Running Time of Each Model in Each Epoch on the Three Datasets

Epoch time(s)	HAN	HeCo	HGCL	SGL	NCL	HAPLRec
Movielens	2.07	0.82	1.16	0.71	0.78	0.70
LastFM	12.76	7.80	3.22	0.98	0.97	1.25
Yelp	84.92	29.00	19.63	3.20	2.15	4.10

Table 6. Different Parameter Selection Strategies

Dataset	Movielens		LastFM		Yelp	
Method	Recall	NDCG	Recall	NDCG	Recall	NDCG
Fixed param	0.3549	0.4140	0.2566	0.2400	0.1101	0.0673
Grid search	0.3609	0.4245	0.2677	0.2568	0.1119	0.0684
APL	**0.3645**	**0.4276**	**0.2842**	**0.2758**	**0.1142**	**0.0690**

less dependent on heterogeneous information, so it is relatively less affected, resulting in a lower percentage of performance degradation. In addition, on the Yelp data set, as the noise ratio increases, while the performance of LightGCN decreases steadily, the performance decrease trend of HAPLRec gradually becomes smaller and tends to be stable. This proves that HAPLRec is more robust on sparse datasets.

5.5 Time Performance Comparison

In order to compare the time performance, we selected several heterogeneous methods HAN, Heco, HGCL and two more advanced contrastive learning recommendation methods SGL and NCL for comparison. The time spent in each epoch of each model is shown in Table 5. Due to the small size of the Movielens dataset, the differences between several methods are not large. On the LastFM and Yelp data sets, HAPLRec has obvious advantages compared with several heterogeneous methods. But compared with SGL and NCL, HAPLRec takes more time per epoch because we introduce more auxiliary information for training. Despite this, our model is also competitive to a certain extent, and the difference with SGL and NCL is not particularly large.

5.6 In-Depth Study of HAPLRec

5.6.1 Effect of APL. To verify the effectiveness of our adaptive preference learning algorithm, we conduct experiments on three datasets for three parameter selection strategies. Among them, the fixed parameters use the same initial coefficients as our adaptive preference learning algorithm, that is, $\lambda = 0.01$, and the grid search is between [0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1] for tuning. It can be seen from Table 6 of the experimental results that under the same initial coefficients, the performance of our dynamic tuning algorithm far exceeds that of fixed parameters. Even with grid search, it is difficult to achieve dynamically tuned performance. Therefore, our adaptive preference learning algorithm can not only save a lot of tuning time, but also achieve very good results.

5.6.2 Effect of Meta-paths. To explore the effect of meta-paths on model performance, we design two variants of the model HAPLRec, as shown in the Table 7. Among them, HAPLRec-u means removing the meta-paths on the user-side, and HAPLRec-i means removing the meta-paths on the item side. According to the results in Table 7, we can find that both HAPLRec-i and HAPLRec-u are better than LightGCN, which indicates that both user relationship graphs and item relationship

Table 7. The Effect of Meta-paths

Dataset	Movielens		LastFM		Yelp	
Method	Recall	NDCG	Recall	NDCG	Recall	NDCG
LightGCN	0.3536	0.4151	0.2743	0.2658	0.1046	0.0666
HAPLRec-u	0.3646	0.4263	0.2818	0.2746	0.1124	0.0681
HAPLRec-i	0.3655	0.4286	0.2826	0.2756	0.1134	0.0685
HAPLRec	**0.3680**	**0.4312**	**0.2864**	**0.2791**	**0.1142**	**0.0690**

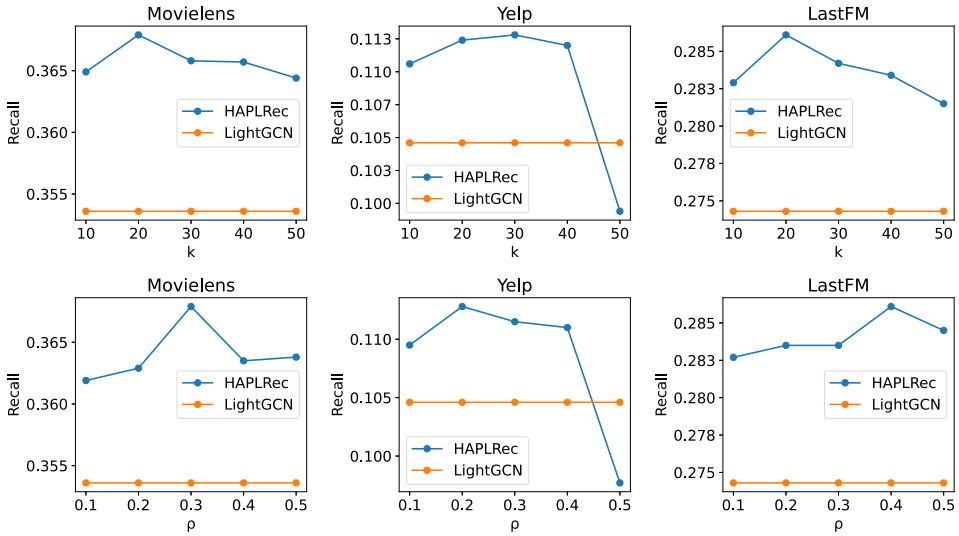


Fig. 6. Performance comparison w.r.t. different k and ρ.

graphs obtained through meta-paths can improve recommendation performance. The model HAPLRec uses both the user-side meta-paths and the item-side meta-paths, and the recommendation performance is further improved, which further proves the effectiveness of the meta-paths.

5.6.3 Impact of the Number of Neighbors k. We tested the performance of the model with different number of neighbors k in [10, 20, 30, 40, 50] on three datasets. The result is shown in Figure 6. We can observe that as k increases from 10 to 50, its performance first increases and then decreases on all three datasets. We think this is because when k is small, the relationship between users and the relationship between items cannot be fully exploited and utilized. When k is large, a large number of noise relations will be introduced, which will affect the training and learning of the model. Especially on the Yelp dataset, we found that when k=50, HAPLRec performs even worse than LightGCN. We guess that because Yelp has more users and items than Movielens and LastFM datasets, it will introduce more noise.

5.6.4 Impact of Drop Ratio ρ. We test the model performance under different drop ratio ρ on three datasets. The result is shown in Figure 6. We can see that HAPLRec is not very sensitive to ρ. We guess this is due to the adaptive preference learning algorithm we adopted, that is, whether it is a larger or smaller ρ, we can use this strategy to make the auxiliary task more conducive to the training of the main task, so we can get more stable results.

6 CONCLUSION AND FUTURE DIRECTIONS

In this work, we recognized the limitation that current contrastive learning recommendation tasks ignored heterogeneous auxiliary data, and explored how to exploit heterogeneous data to address this limitation. In particular, we proposed a framework, HAPLRec, to obtain user and item relationship graphs on heterogeneous graphs using a meta-path sampling-based method, and used them to construct contrastive learning auxiliary tasks. In addition, we explored the coefficient setting of auxiliary tasks in multi-task learning, and proposed the adaptive preference learning algorithm, which dynamically adjusts the coefficients of auxiliary tasks during the training process. We conducted extensive experiments on three benchmark datasets to demonstrate the effectiveness of our proposed method. We also validated the ability of HAPLRec to alleviate data sparsity and noise problems.

As future work, we will investigate how to more fully utilize heterogeneous auxiliary data. In this work, we ignored other types of nodes in the process of constructing user relationship graphs and item relationship graphs through meta-paths. Therefore, how to utilize other types of nodes is worth exploring. In addition, a common problem in heterogeneous recommendation is the selection of meta-paths. In this work, we manually set meta-paths through experience. How to make the model automatically select meta-paths is also a direction of our future work.

REFERENCES

[1] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[2] Yuanjiang Cao, Xiaocong Chen, Lina Yao, Xianzhi Wang, and Wei Emma Zhang. 2020. Adversarial attacks and detection on reinforcement learning-based interactive recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1669–1672.

[3] Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, Yong Xu, and Ronghua Luo. 2023. Heterogeneous graph contrastive learning for recommendation. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining*. 544–552.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*. PMLR, 1597–1607.

[5] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*. PMLR, 794–803.

[6] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. 2020. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems* 33 (2020), 2039–2050.

[7] Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796* (2020).

[8] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 6894–6910.

[9] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[10] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 1725–1731.

[11] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. 2018. Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision*. 270–287.

[12] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 297–304.

[13] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 1923–1933.

[14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 639–648.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

[16] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 549–558.

[17] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. 2018. Leveraging meta-path based context for top-N recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1531–1540.

[18] Adrián Javaloy and Isabel Valera. 2022. RotoGrad: Gradient homogenization in multitask learning. In *Proceedings of the 10th International Conference on Learning Representations*.

[19] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7482–7491.

[20] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[22] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

[23] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*. 689–698.

[24] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *Proceedings of the ACM Web Conference 2022*. 2320–2329.

[25] Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. 2021. Towards impartial multi-task learning. In *Proceedings of the 9th International Conference on Learning Representations*.

[26] 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 *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1160–1169.

[27] Chen Luo, Wei Pang, Zhe Wang, and Chenghua Lin. 2014. Hete-CF: Social-based collaborative filtering recommendation using heterogeneous relations. In *2014 IEEE International Conference on Data Mining*. IEEE, 917–922.

[28] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. 2008. SoRec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. 931–940.

[29] Yunshan Ma, Yingzhi He, An Zhang, Xiang Wang, and Tat-Seng Chua. 2022. CrossCBR: Cross-view contrastive learning for bundle recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1233–1241.

[30] Itzik Malkiel and Lior Wolf. 2021. MTAdam: Automatic balancing of multiple training loss terms. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 10713–10729.

[31] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[32] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1150–1160.

[33] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. 2022. Contrastive learning for representation degeneration problem in sequential recommendation. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*. 813–823.

[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. 452–461.

[35] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[36] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. 285–295.

[37] Ozan Sener and Vladlen Koltun. 2018. Multi-task learning as multi-objective optimization. *Advances in Neural Information Processing Systems* 31 (2018), 525–536.

[38] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and Philip S. Yu. 2018. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 31, 2 (2018), 357–370.
[39] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.
[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017).
[41] Nelson Vithayathil Varghese and Qusay H. Mahmoud. 2020. A survey of multi-task deep reinforcement learning. *Electronics* 9, 9 (2020), 1363.
[42] Siyu Wang, Xiaocong Chen, Dietmar Jannach, and Lina Yao. 2023. Causal decision transformer for recommender systems via offline reinforcement learning. *arXiv preprint arXiv:2304.07920* (2023).
[43] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 165–174.
[44] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.
[45] Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. 2021. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1726–1736.
[46] Xin Wang, Wenwu Zhu, and Chenghao Liu. 2019. Social recommendation with optimal limited attention. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1518–1527.
[47] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. 2021. Contrastive learning for cold-start recommendation. In *ACM Multimedia*. 5382–5390.
[48] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 726–735.
[49] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: A survey. *ACM Computing Surveys (CSUR)* 55, 5 (2022), 1–37.
[50] Xin Xia, Hongzhi Yin, Junliang Yu, Yingxia Shao, and Lizhen Cui. 2021. Self-supervised graph co-training for session-based recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2180–2190.
[51] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2021. Self-supervised hypergraph convolutional networks for session-based recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4503–4511.
[52] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Guandong Xu, and Quoc Viet Hung Nguyen. 2022. On-device next-item recommendation with self-supervised knowledge distillation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 546–555.
[53] Ruobing Xie, Qi Liu, Liangdong Wang, Shukai Liu, Bo Zhang, and Leyu Lin. 2022. Contrastive cross-domain recommendation in matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4226–4236.
[54] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th International Conference on Data Engineering*. IEEE, 1259–1273.
[55] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Ting Chen, Aditya Menon, Lichan Hong, Ed H. Chi, Steve Tjoa, Jieqi Kang, and Evan Ettinger. 2021. Self-supervised learning for large-scale item recommendations. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4321–4330.
[56] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.
[57] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems* 33 (2020), 5812–5823.
[58] Junliang Yu, Min Gao, Jundong Li, Hongzhi Yin, and Huan Liu. 2018. Adaptive implicit friends identification over heterogeneous network for social recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 357–366.
[59] Junliang Yu, Hongzhi Yin, Min Gao, Xin Xia, Xiangliang Zhang, and Nguyen Quoc Viet Hung. 2021. Socially-aware self-supervised tri-training for recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2084–2092.

[60] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In *Proceedings of the Web Conference 2021*. 413–424.

[61] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. 2024. Self-supervised learning for recommender systems: A survey. *IEEE Transactions on Knowledge and Data Engineering* 36, 1 (2024), 335–355.

[62] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems* 33 (2020), 5824–5836.

[63] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. 2013. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the 7th ACM Conference on Recommender Systems*. 347–350.

[64] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete collaborative filtering. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 325–334.

[65] Junwei Zhang, Min Gao, Junliang Yu, Lei Guo, Jundong Li, and Hongzhi Yin. 2021. Double-scale self-supervised hypergraph learning for group recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2557–2567.

[66] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[67] Wenlu Zhang, Rongjian Li, Tao Zeng, Qian Sun, Sudhir Kumar, Jieping Ye, and Shuiwang Ji. 2015. Deep model based transfer and multi-task learning for biological image analysis. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1475–1484.

[68] 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.

[69] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2021. RecBole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4653–4664.

[70] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive learning for debiased candidate generation in large-scale recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3985–3995.

[71] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1893–1902.

Received 29 August 2023; revised 4 March 2024; accepted 4 March 2024