Complex & Intelligent Systems (2025) 11:153
https://doi.org/10.1007/s40747-024-01778-5

ORIGINAL ARTICLE

DMR: disentangled and denoised learning for multi-behavior recommendation

Yijia Zhang[1] · **Wanyu Chen**[1] · **Fei Cai**[2] · **Zhenkun Shi**[1] · **Feng Qi**[3]

Received: 29 August 2024 / Accepted: 29 December 2024 / Published online: 16 January 2025

Abstract

In recommender systems, leveraging auxiliary behaviors (e.g. view, cart) to enhance the recommendation in the target behavior (e.g. purchase) is crucial for mitigating the sparsity issue inherent in single-behavior recommendation. This has given rise to the multi-behavior recommendation (MBR). Existing MBR task faces two primary challenges. First, the irrelevant auxiliary behaviors that do not align with the target behavior, can negatively impact the prediction accuracy for user preference in the target behavior. Second, these methods typically learn coarse-grained user preferences, failing to model the consistency and distinctiveness among multiple behaviors at a fine-grained level. To address these issues, we propose a disentangled and denoised model for multi-behavior recommendation (DMR), which employs user preferences reflected in the target behavior to guide the learning of user and item embeddings in auxiliary behaviors. Specifically, we first design a disentangled graph convolutional network, modeling the fine-grained user preference under multiple behaviors in view of item attribute domains. We also propose a denoised contrastive learning strategy, where we align the user preferences in multiple behaviors by reducing the influence of noisy data existing in auxiliary behaviors. Experimental results on two real-world datasets show the proposal can improve the performance of MBR models effectively, which achieves on average 3.12% on the Retailrocket dataset and 3.28% on the Beibei dataset over the performance of state-of-the-art baselines. Extensive experiments also demonstrate our model's competitive performance for fine-grained preference learning and denoised learning.

Keywords Multi-behavior recommendation · Fine-grained preferences · Contrastive learning · Graph convolutional network

Introduction

Recommender system has been an essential tool in daily life due to its effectiveness for alleviating information overloading [1, 2]. Collaborative filtering (CF) has remained the most extensively utilized algorithm in recommender systems, and there exists plenty of CF-based models utilizing traditional matrix factorization [3] and deep neural network techniques [4–6], these methods achieve significant success in providing valuable recommendations that guide users in discovering their genuine interests. However, most CF-based methods are designed around a specific user behavior, which is contrary to the different behavior patterns in real-world recommendation scenarios. For example, in the e-commerce platform, most CF-based models only consider purchase behavior for predicting users' preferences, while neglecting other behaviors such as view and cart, these behaviors also partly reflect the user's purchase intent and play an important role in alleviating the sparsity problem. The limitations of considering single user behavior motivates studies on exploiting the multiple types of user behaviors, making use of the auxiliary behaviors (e.g. view, cart) to enhance the prediction of users' preferences regarding the target behavior (e.g. purchase) [7–10], which is the multi-behavior recommendation (MBR).

Existing MBR models mainly employ the graph convolutional network (GCN) technique to model semantic interactions between users and items across multiple behaviors [11, 12], due to the superior ability of GCN-based methods in modeling graph structure data [13–15]. Specifically, these GCN-based methods model user-item interactions under

✉ Wanyu Chen
wanyuchen@nudt.edu.cn

1 College of Electronic Countermeasures, National University of Defense Technology, No. 460 Huangshan Road, Shushan District, Hefei, Anhui, China

2 College of Systems Engineering, National University of Defense Technology, Changsha, China

3 Tianjin Institute of Industrial Biotechnology, Chinese Academy of Sciences, Tianjin, China

each behavior through the propagation of both user and item embeddings [12, 16–19], then obtain the aggregated user and item embeddings from different behaviors to estimate the user preference in the target behavior. Recently, with the widespread use of the contrastive learning (CL) for data augmentation in the recommendation system, there is an increasing number of studies integrating CL and GCN techniques to deal with the MBR task [17]. These methods apply GCN to learn user and item embeddings and exploit CL to further align the user preference in the target behavior and auxiliary behaviors, the CL can complement GCN to enhance the consistent learning of user preference in multiple behaviors [20–22]. Typical models such as S-MBRec (self-supervised graph neural networks for multi-behavior recommendation) utilizes several parallel GCNs to model different types of user-item interactions, then designs the contrastive learning to alleviate the impact of noise data in auxiliary behaviors, the method shows superior improvements than models that solely rely on GCN. Even though existing methods have achieved significant success in the MBR task, they still confront several challenges as detailed subsequently.

These methods do not take into account fine-grained user preferences regarding specific item characteristics. In real-world recommendation scenarios, user behaviors such as view, cart, and purchase occur primarily because they are attracted to specific attributes of the item. As Fig. 1 shows, the user has purchased a T-shirt with color as blue and material as cotton, also views and carts a series of items that have similar attributes to the purchased T-shirt. Obviously, the user preference is not simply reflected in interacted items, but is focused more attention on specific item attributes, and the correlations between different user behaviors are also reflected in fine-grained item attributes. Existing GCN and CL based MBR models only consider user preferences based on similarities between user and item embeddings, which will lead to limitations for capturing the user's accurate interest in some cases [23–25]. For example, these methods can't capture fine-grained item attributes, thus may recommend a pink dress for the user, while the user's actual intent is a blue and cotton dress. Some previous work has demonstrated the actual preferences of the user may be more complex including multiple dimensions [26–28]. Therefore, we regard that it is necessary to model and align user preference reflected in their behaviors from the perspective of fine-grained item attributes.

They ignore the noise information at the level of item attributes in auxiliary behaviors. The noise data in the MBR task refers to the data in auxiliary behaviors that is inconsistent with the preference distribution in the target behavior. Before discovering the true purchase intent, a user may view or cart items that are presented by the recommender system. Those items may attract his/her attention due to their popularity or advertising which do not correspond to the user's actual purchase intent, thus providing noise information that are detrimental for predicting the accurate user's target intent [21, 29]. These noise data are mixed with other valid data, and most recommendation models are unable to explicitly distinguish these data [30]. As a result, how to automatically remove noise data from auxiliary behaviors has become a challenge. Existing CL based models can reduce the redundant information in auxiliary behaviors through aligning user embeddings in auxiliary behaviors and the target behavior [17]. However, the ability of denoising learning through entangled user embeddings is limited due to the fine-grained user preference towards item attributes. For example, from Fig. 1 we can observe that the user's view behavior includes a pink dress, while it doesn't mean the user wants to buy a pink dress, the user views it is probably due to the style and material of the dress. In this regard, the noise data in auxiliary behaviors mainly manifests in some item attributes that are not related to the user preference in the target behavior. Therefore it is necessary to understand the consistency and distinctiveness of user preferences in multiple behaviors, reducing the noise data in a more fine-grained way.

Based on the above discussions, the user usually focuses on specific item attributes, and the correlations and distinctiveness among multiple behaviors are also based on item attributes. Towards this end, we propose a novel disentangled and denoised model for MBR, which is shorted called DMR. Generally, the model learns fine-grained user and item embeddings through a disentangled graph convolutional network (DGCN), and plays a role in denoising mainly through target behavior guided weight in DGCN and multi-attributes linearized attention mechanism (MLAM) in CL. To be specific: First, we inject item embeddings into several attribute domains to disentangle user and item embeddings, which can model the user's preference towards specific item attributes in the latent semantic space. Then, based on the disentangled attribute domains, we propose DGCN to learn the user preference in view of different item attribute domains under each behavior. In each attribute domain for auxiliary behaviors, we design target behavior guided weight to guide the learning of weights for user-item interactions in auxiliary behaviors. This is used for capturing the attributes for items in auxiliary behaviors that are closely related to user preferences in the target behavior. We incorporate the weights calculated into the propagation layers of the GCN to obtain the ultimate user and item embeddings in each item attribute domain. Next, we design a denoised contrastive learning (DCL) strategy to align user preferences in auxiliary behaviors and target behavior, we design MLAM to further deal with the noise data in auxiliary behaviors. We explore the importance of different attribute domains in auxiliary behaviors toward user intent in the target behavior, which can be used for contrastive learning in a denoised way so as to distinguish positive and

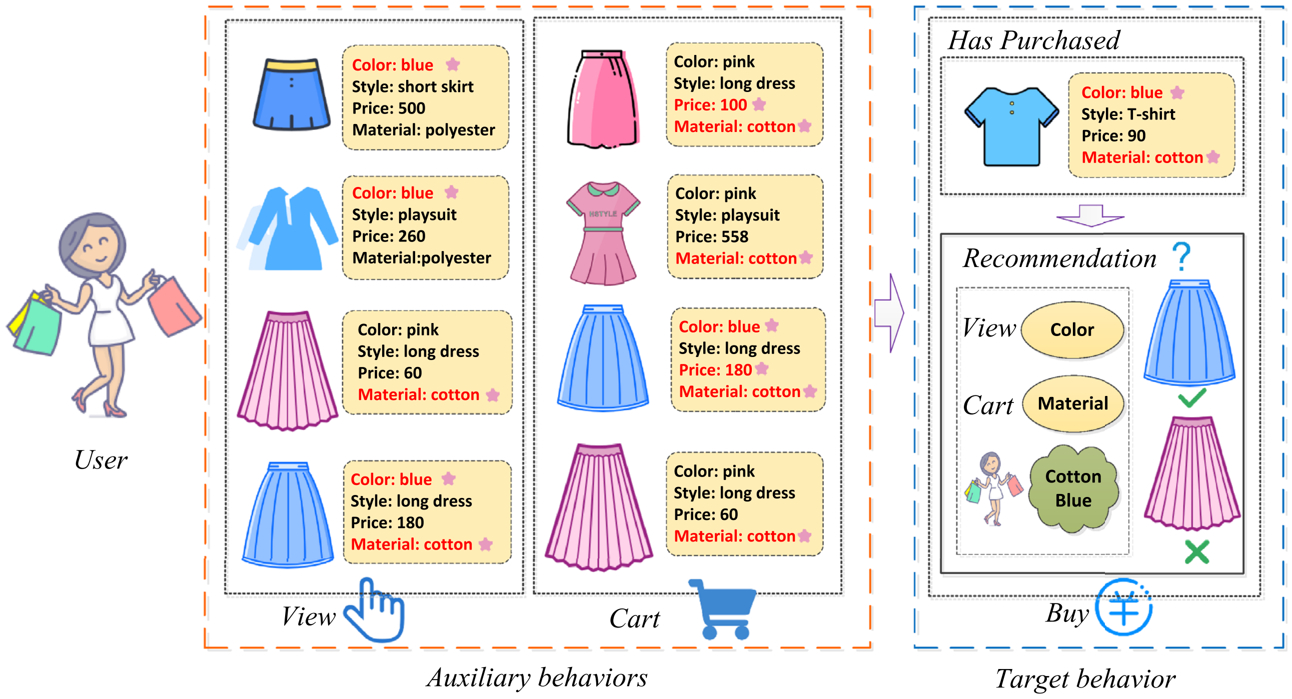

Fig. 1 The highlighted part represents the same attributes of items that have been interacted with by users in Cart and view behaviors, as well as items that have been interacted with in target behavior. The model learns that users pay more attention to color attributes (blue) and material attributes (cotton). The final prediction result is a blue and cotton dress, instead of a red dress

negative samples thoroughly. Comprehensive experiments on two real-world datasets validate the effectiveness of our model. The results show that our model outperforms the state-of-the-art baselines. We also conduct ablation studies to demonstrate the contributions of each module in our model in terms of improving the model performance on the MBR task. In brief, our main contributions are as follows:

- First, we disentangle item representations to obtain several different item attribute domains for the modeling of fine-grained user preference in multiple behaviors. To the best of our knowledge, our model is the first study to learn disentangled user preferences in multiple behaviors from the perspective of item attributes.
- Second, we propose the DGCN to capture attributes of the item that the user interacted with in auxiliary behaviors, which are more in line with the user's purchase intent, thus denoising the noise attributes for items in auxiliary behaviors.
- Third, we design the DCL to align the user preference in the target behavior and auxiliary behaviors, which can enhance the learning of correlations between auxiliary behaviors and target behavior towards different attribute domains.
- Finally, we conduct experiments on two real datasets and compare them with some advanced methods to demonstrate that our proposed method can effectively improve the performance of MBR. The extensive experiments also validate the abilities of our model in denoising learning.

Related work

Multi-behavior recommendation

The practice of multi-behavior recommendation involves capitalizing on various forms of user-item interactions to augment the effectiveness of the recommendation. It typically aims to refine the prediction accuracy for the target behavior by distilling valuable information from auxiliary behavior data. Early works are most relied on conventional collaborative filtering techniques, including matrix factorization (MF) and Bayesian personalized ranking (BPR) [31]. For example, Singh et al. proposed CMF (collective matrix factorization [7]) to model different user-item interaction relations, which jointly factorize multiple behavior matrices. Meanwhile, some researchers attempt to extend the BPR to the multi-behavior recommendation task such as MC-BPR (multi-channel Bayesian personalized ranking [32]), it designs negative sampling strategies by generating samples from multiple behaviors.

Recently, the graph convolutional network has been widely applied in the multi-behavior recommendation task [11, 12, 17, 33–35] due to its strong ability for learning features from heterogeneous user-item interactions. For example, Jin et al. proposed MBGCN (multi-behavior graph convolution network [11]), which utilizes GCN to learn behavior strength and semantics. Then, Chen et al. proposed GHCF (graph heterogeneous collaborative filtering [33]), which also utilizes GCN to learn user preference for multiple behaviors. Specifically, GHCF further embed the different relations of user-item interactions, incorporating both user/item and relation embedding for learning the user preference under multiple behaviors. Meanwhile, Xia et al. proposed MB-GMN (graph meta network for multi-behavior recommendation [12]), they adapt meta-learning to capturing semantic transfer relationships between different behaviors, which improves the ability of GCN for mining different user behavior patterns. Later, Gu et al. proposed S-MBRec (self-supervised graph neural networks for multi-behavior recommendation [17]) by integrating self-supervised methods. They constructed several independent user-item bipartite graph for different behaviors instead of constructing a unified heterogeneous graph, and they designed a novel contrastive learning to model the embedding commonality among different behaviors. Latest, He et al. proposed CIGF (compressed interaction graph for multi-behavior recommendation [19]) through the innovative learning of a compressed interaction graph convolution network. This network is specifically designed to explicitly capture and model high-order relations, thereby refining the representation learning process for users and items.

Based on the above discussion, existing multi-behavior based recommendation methods are mainly based on GCN and CL. They utilize GCN to capture user and item feature representation through user-item interactions of different behaviors, and combine multiple behavior information to learn the final user preferences. In addition, CL method is used in these methods align different user behavior preferences by considering different user behaviors as different contrastive views. Though these methods have achieved remarkable success in the multi-behavior recommendation task, they can't capture the fine-grained user preference under different behaviors, thus ignoring the fine-grained connections and differences among different behaviors.

Graph-based recommendation

Recently, architectures built on diverse variations of graph neural networks (GNN) have exhibited trailblazing performance for multiple graph-structured data based tasks [36–39], there emerging a great number of GNN-based model in the recommendation domain [5, 40–49]. For example, Berg et al. earlier proposed a graph auto-encoder framework named GCMC (graph convolutional matrix completion [41]), this framework innovatively addresses the rating prediction challenge in recommendation systems by adopting a link prediction approach. Later, He et al. proposed several GCN-based methods for the recommendation domain [5, 42, 45, 50]. They first proposed NGCF (neural graph collaborative filtering [5]) to exploit injecting the collaborative information between user items into the embedding representation of users and items learned by GCN, and they utilized high-order connectivity of user item interactions to build the neural graph collaborative filtering framework. Then, they further proposed LightGCN [50] by reducing several unnecessary and complex parts of the typical GCN methods, which obtains the final embeddings by simply propagating user and item embeddings linearly. In addition, some works utilized the GAT (graph attention network [51]) mechanism to improve the aggregation operations [45–47]. For example, Wang et al. proposed KGAT (knowledge graph attention network [45]), which identifies the importance of different neighbors through the attention mechanism in KG, this work also models the high-order connectivity.

Inspired by the significant performance of GNN in the recommendation task, we improve the existing GCN-based multi-behavior recommendations by designing the disentangled graph attention network, which can learn more fine-grained user preference than existing GCN-based methods, meanwhile capturing more useful information for learning user preference in the target behavior from auxiliary behaviors.

Contrastive learning based recommendation

Contrastive learning (CL) aims to enhance the consensus among various perspectives, with the goal of acquiring superior representations through self-supervision. Recently, CL has achieved consequential improvements in various fields [52–58], and some recent works have illustrated that CL can improve the recommendation performance remarkable [59–66]. For example, Yu et al. [59] observed that graph augmentations are not necessary for the recommendation tasks, they innovatively replaced the graph augmentations with the uniform noises added for constructing contrastive views, achieving superior experimental results than the CL methods based on graph augmentations. Later, Lin et al. proposed NCL (neighborhood-enriched contrastive learning [60]) by integrating CL into the graph embedding learning according to the graph structure. In this work, the positive pairs are the structural neighbors of users (items), which aims to enhance the graph representation learning by integrating the structural information. Meanwhile, Xia et al. proposed HCCF (hypergraph contrastive collaborative filtering [61]), they first learned a hypergraph structure to reflect dependency relationships, then they introduced a new hypergraph contrastive learning architecture with complementary self extracting views, which can effectively solves the over-smoothing problem of the GCN. Recently, Cai et al. [62] proposed lightGCL, they utilized SVD (single value decomposition) method to generate the contrastive augmentation views, instead of using the existing augmentation methods such as random enhancement and heuristic enhancement, alleviating issues that negatively impact the generality and robustness of contrastive learning based recommendations.

Existing CL based methods mainly align user preferences in a coarse-grained level. Different from them, we denoise the user embeddings in each attribute domains, and aggregate the denoised user embeddings in different attribute domains, thus obtaining the finer-grained user embeddings for the contrastive learning.

Methodology

Problem definition

Let $\mathcal{U}$ stand for the users set, $\mathcal{I}$ symbolize items set, with $|\mathcal{U}|$ and $|\mathcal{I}|$ characterizing the number of users and items respectively. We define the bipartite graph $G = (V, E)$ according to the user-item interactions, where $V = U \cup I$. In scenarios involving multiple behaviors, we consider $K(K \geq 2)$ distinct user-item behaviors, with edges corresponding to the $k_{th}(1 \leq k \leq K)$ behavior as E_k. The bipartite G is partitioned into K subgraphs $G_1, G_2, \ldots, G_K$, each aligned with a specific interaction behavior type and formulated as $G_k = (V, E_k)$. $G_1, G_2, \ldots, G_{K-1}$ denotes subgraphs of auxiliary behaviors, G_K denotes subgraph of the target behavior. These behavioral subgraphs $G_1, G_2, \ldots, G_K$ are represented by interaction matrices $Y_1, Y_2, \ldots, Y_K$, interaction matrix Y_k is the binary form defined following implicit feedback recommendation [4]. We summarize the notations used in this paper in Table 1.

$$y_{ui}^{k} = \begin{cases} 1, & \text{if } u \text{ has interacted with } i \text{ under behavior } k; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Our purpose in this paper is to estimate the potential for user-item interactions with respect to the target behavior using multiple behaviors. The model utilizes historical user-item interactions in both auxiliary and target behaviors as the input, and output the likelihood that a user u will interact with an item i under the target behavior. Finally, the model performs recommendations under the target behavior according to the likelihood. The task of our model is formulated as:

Input: The multi-behavior interaction matrices $Y_1, Y_2, \ldots, Y_K$ under multiple K types of behaviors.

Output: A model that estimates the likelihood that a user u will interact with an item i under the target behavior.

Proposed model

In this paper, we propose a disentangled and denoised model for MBR, the architecture of this work is depicted in Fig. 2. The user-item interaction matrices under different behaviors are used as input for the model, and user and item embeddings from different behaviors are concatenated to represent the ultimate user and item embeddings. To achieve this goal, the model incorporates the meticulous design of the following modules. (i) Disentangled embedding generating. To model the fine-grained user preference from the perspective of item attributes, we first inject item embeddings into several disentangled domains. We also design the constraint function to ensure the independence of different attribute domains and prevent them from degenerating into a single attribute space. Afterward, we obtain the initial disentangled user and item embeddings in each attribute domain for multiple behaviors. (ii) Disentangled graph convolutional network. In each attribute domain, we first utilize a user's general preference in the target behavior to guide the learning of the aggregation weights of user-item interactions in auxiliary behaviors, learning the attractiveness of different items to the user for specific attributes in auxiliary behaviors. Then, we incorporate these weights into the GCN for the aggregation of user (item) nodes' neighbors to obtain user (item) embeddings in view of the attribute domain. Finally, we aggregate user and item embeddings in different attribute domains to obtain these embeddings under multiple behaviors. Through DGCN, we can discover the item attributes that the user is truly interested in and alleviate the noise information introduced by items in auxiliary behaviors. (iii) Denoised contrastive learning. For user and item embeddings in different attribute domains under multiple behaviors, we first employ the attention mechanism to estimate the importance of different attribute domains in auxiliary behaviors, we assign a higher score for the user embedding in a certain attribute domain when this attribute in auxiliary behaviors achieves more user attention for his/her target intent. Then we aggregate user embeddings for different attribute domains according to these attention scores to generate the denoised embeddings. Finally, we design contrastive learning on these denoised embeddings to align user preferences in auxiliary behaviors and target behavior. The aggregated user embeddings from different attribute domains utilizing attention scores are concatenated to the user embeddings learned through DGCN to achieve the final user embeddings for multiple behaviors.

Table 1 The descriptions of notations

Notation	Description
$\mathcal{U}$	User set
$\mathcal{I}$	Item set
M	Number of users
N	Number of items
L	Number of graph attention network layers
A	Number of item attributes
Q	Embedding matrix for the item embedding initialization
P	Embedding matrix for the user embedding initialization
e_u	Initialized user embedding of user u
e_i	Initialized item embedding of item i
$e_{u,a}$	User embedding of user u for attribute a
$e_{i,a}$	Item embedding of item i for attribute a
e_u^k	User embedding of user u under the behavior k
e_i^k	Item embedding of item i under the behavior k
q_u	Final user embedding of user u
q_i	Final item embedding of item i
$q_{u,a}$	Query vector of user u for attribute a
$e_{i,n}^{u,a}$	Item embedding of n_{th} item that have interacted with user u for attribute a
W_Q, W_K, W_V	Weight matrices of the self attention
W_a	Weight matrix of the target behavior guided attention
W_b	Weight matrix of the item embedding decomposed
M_u^a	Projection matrix of user $u's$ embedding initialization for the attribute a
W_u	Weight matrix of user u to obtain the final user embedding
W_i	Weight matrix of item i to obtain the final item embedding

Disentangled embedding generating

As discussed in Sect. 1, the correlations between user preferences in different behaviors are mainly reflected in different item attributes, and the noise data in auxiliary behaviors is also intimately associated with item attributes. Therefore, it is suitable to focus on the specific item attribute than the entire item embedding without emphasis. Specifically, we first obtain the initialized user and item embeddings, then we disentangle them to generate the user and item embeddings in each attribute domain.

User and item embedding initialization Consistent with prevalent initialization methods used in existing recommendation methods [67], we connect each user and item to their respective identify embeddings. We utilize $P \in \mathbb{R}^{M\times D}$ and $Q \in \mathrm{R}^{N\times D}$ to represent the initialized embeddings of user's as well as items respectively, where M denotes the number of users, N denotes the number of items, d denotes the embedding size. In addition, D_u^U and D_i^I are one-hot vectors for the user u and item i, the initialized user and item embeddings are denoted as below:

$$\begin{aligned} e_u &= P^T D_u^U \\ e_i &= Q^T D_i^I \end{aligned} \tag{2}$$

where T denotes the transposition operation, e_u is the initialized user embedding of the user u, e_i is the initialized item embedding of the item i.

Disentangled embedding generating In this work, we argue that the user's preference for items is mainly reflected in their preference for specific item attributes. However, item attributes in different recommendation scenarios are complex and diverse, making it difficult to model unified attribute representations. Therefore, we construct several disentangled attribute domains to represent item attributes in the latent semantic space. Inspired by the work [68], we can obtain the item embeddings by decomposing the initial item embedding for each attribute domain. Assuming that A is the set of attributes that $A = \{1, 2, \ldots, |A|\}$ and $|A|$ is the number of attribute domains, we project the item embedding e_i of item i into the item embedding for attribute $a \in A$ by employing the projection matrix as below:

$$e_{i,a} = \frac{W_b e_i}{\|W_b e_i\|_2} \tag{3}$$

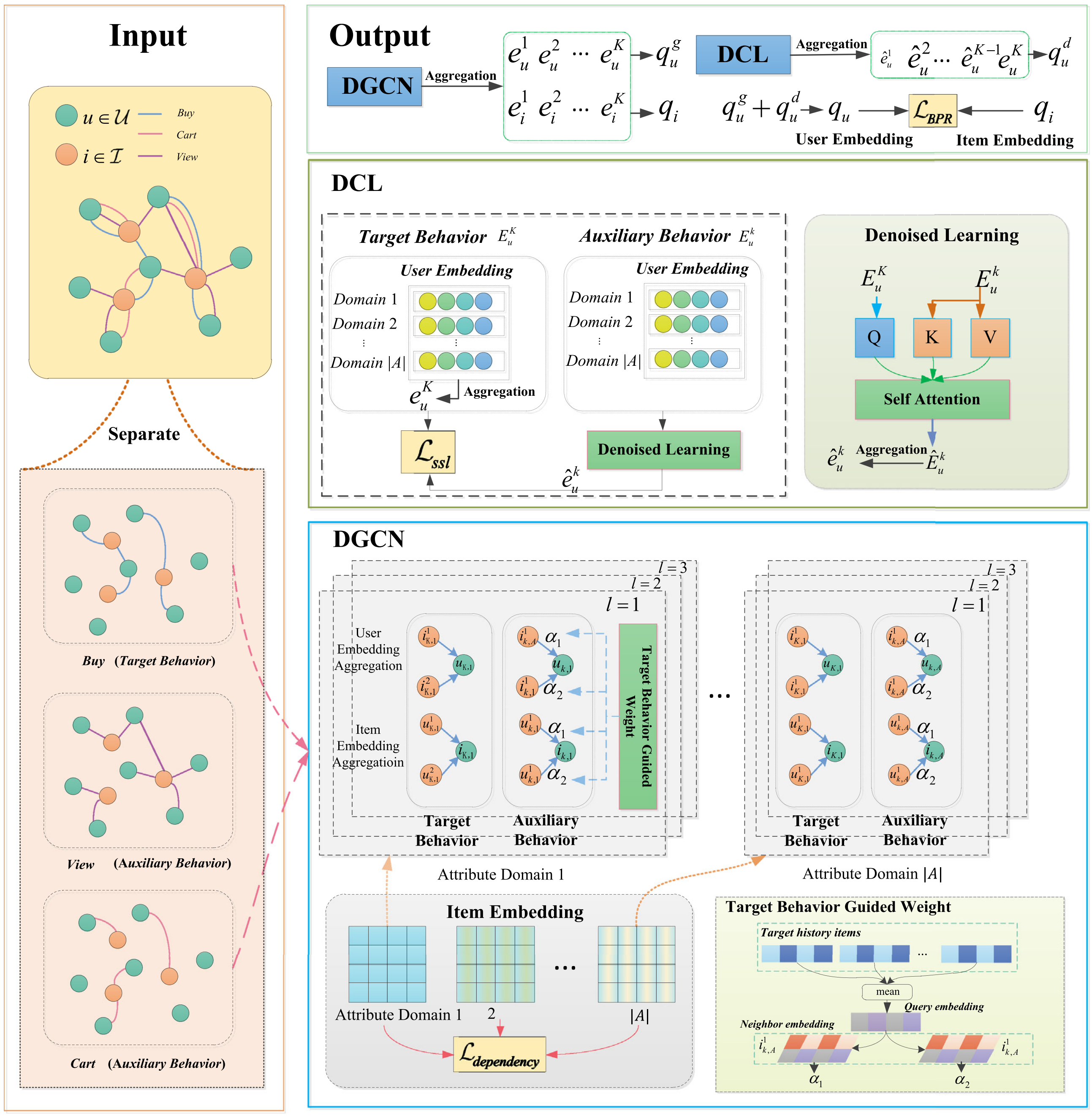

Fig. 2 The architecture of our model

where $e_{i,a} \in \mathbb{R}^d$ is the embedding of the item i in the attribute domain a, $W_b \in \mathbb{R}^{d \times d}$ denotes a projection matrix of the attribute domain a, noted that the projection matrix is shared by all items. Additional, the normalization for the item embeddings is to ensure the subsequent calculations.

We encourage the separation of item embeddings in different attribute domains due to different attribute domains should contain different information about item attributes, otherwise several item embeddings will degenerate to the equivalent of a single item embedding [69]. Hence, for superior model capacity and explainability, We incorporate an independency loss to constrain the item embeddings for different attribute domains. Following the existing methods such as [69], we adopt mutual information to measure the correlations between two item embeddings. For each item i, the constraint function for embeddings independence is designed as below:

$$\mathcal{L}_{ind}^{i} = \sum_{b \in A} - \log \frac{\exp\left(\frac{s(e_{i,a}, e_{i,a})}{\tau}\right)}{\sum_{a' \in A} \exp\left(\frac{s(e_{i,a}, e_{i,a'})}{\tau}\right)} \tag{4}$$

where $s(\cdot)$ is the function measuring the similarity of two item embeddings in different attribute domains, τ is the temperature parameter, $s(\cdot)$ is set as a cosine similarity function as below:

$$s(e_1, e_2) = \frac{e_1 e_2^T}{\|e_1\|_2 \|e_2\|_2}. \tag{5}$$

The final dependency loss are composed of the mutual scores of all the items is as below:

$$\mathcal{L}_{ind} = \sum_{i \in \mathcal{I}} \mathcal{L}_{ind}^{i}. \tag{6}$$

After decomposing the item embeddings, we perform initial representations of user representations in each attribute domain in order to learn user embeddings regarding to different item attributes. For each behavior $k (k \leq K)$, we utilize the transformation matrix $M_u^a \in \mathbb{R}^{a \times d \times d}$ to obtain the user embedding as the initial user embedding for the GCN under each behavior k as below:

$$e_{u,a}^{k} = M_u^a e_u \tag{7}$$

where $e_{u,a}^{k} \in \mathrm{R}^{a \times d}$ is the initial user embeddings with respect to the attribute domain a under behavior k, it is also as an initial representation of the subsequent graph network learning. The initial embedding of the item for the GCN under each behavior k is as below:

$$e_{i,a}^{k} = e_{i,a}. \tag{8}$$

Disentangled graph convolutional network

After obtaining the initial user and item embeddings in different attribute domains, we then develop a disentangled graph learning-based framework to model the fine-grained user preference for specific item attributes under multiple behaviors. We perform a GCN in each view of attribute domain under each behavior, meanwhile selecting the attributes of items that are consistent with the user preference in the target behavior. Specifically, we first calculate the target-behavior guided weight to estimate the importance of different item attributes that the user interacted with under auxiliary behaviors. This enables the model to concentrate more attention on items engaged in auxiliary behaviors that express closer resemblance to the user preference in the target behavior regarding specific item attributes. Then, we integrate the weight into the GCN to obtain final user and item embeddings, where a weighted message-passing method is designed for the propagation of user and item embeddings. In the propagating of user nodes towards specific attribute domains, if the item embedding in the attribute domain is more closely aligned with the user's target behavior preference, the information passed by the item to the user node should be assigned a higher weight compared to other neighbor item nodes.

Target behavior guided weight To determine the score of user u and its neighbor item node i in the attribute domain a, our first step involves learning a query vector q that could reflect u's general preference reflected in the target behavior. In our pursuit of understanding a user's target behavior preference, we employ the average value of the embeddings of all items that interacted with the user in this attribute domain to obtain a query vector, serving as a robust representation of the user's behavioral preference. The function for calculating the query vector q is here:

$$q_{u,a}^{K} = \mathrm{mean}\{e_{i,1,a}^{u,K}, e_{i,2,a}^{u,K}, \ldots, e_{i,n,a}^{u,K}\} \tag{9}$$

where $q_{u,a}^{K} \in \mathbb{R}^d$ denotes the general preference of user u in attribute domain a under the target behavior K, $e_{i,n,a}^{u,K} \in \mathrm{R}^d$ denotes the embedding of n_{th} item node in the attribute domain a that interacted with the user u under the target behavior K, which is learned by Formula 8. Afterwards, we use the learned general preference representation to calculate the score of the user u and neighbor item i as below:

$$\alpha_{u,i}^{a,k} = \frac{W_a[q_{u,a}^{K}; e_{i,a}^{k}]}{\sum_{j \in \mathcal{N}_u^k} W_a[q_{u,a}^{K}; e_{j,a}^{k}]}, \tag{10}$$

where $\mathcal{N}_u^k$ denotes item neighbors set for the user u under the behavior k, $i, j \in \mathcal{N}_u^k$ denote the neighbor items i and j for the user u under the behavior $k (k < K)$, $e_{i,a}^{k}, e_{j,a}^{k} \in \mathrm{R}^d$ denote embeddings of these items in the attribute domain a under the behavior $k (k < K)$, $W_a \in \mathbb{R}^{d \times d}$ demotes the weight matrix, which is the hyper-parameter of the model, [;] denotes the concatenation of vectors.

Graph convolutional network Once the target behavior-guided weights have been determined, we employ the GCN as most existing MBR models to effectively aggregate the neighbors of users and items in each view of item attribute domain under multiple behaviors. For auxiliary behaviors, the weights mentioned previously are incorporated during the graph information propagation process. We design the aggregation function under the specific auxiliary behavior for user and item nodes as below:

$$\begin{aligned} e_{u,a}^{k,l+1} &= \sum_{i \in \mathcal{N}_u^k} \alpha_{u,i}^{a,k} e_{i,a}^{k,l} \\ e_{i,a}^{k,l+1} &= \sum_{u \in \mathcal{N}_i^k} \alpha_{u,i}^{a,k} e_{u,a}^{k,l} \end{aligned} \tag{11}$$

where $\mathcal{N}_u^k$ and $\mathcal{N}_i^k$ denote the neighbor nodes set of user u and item i under the behavior $k(k < K)$, l denotes the layer number of the GCN, $e_{u,a}^{k,l+1} \in \mathbb{R}^d$ and $e_{i,a}^{k,l+1} \in \mathbb{R}^d$ denote embeddings of the user u and item i in the $(l+1)_{th}$ layer under the behavior $k(k < K)$, $\alpha_{u,i}^{a,k}$ is the weighted obtained through Formula 10.

For the GCN towards the target behavior, the $\alpha_{u,i}^{a,k}$ is assigned as the value 1, which is in line with the LightGCN. For the attribute domains $A = \{1, 2, \ldots, |A|\}$ under the behavior k. Finally, the user and item embeddings in view of attribute domains are derived from the last layer L of GCN, denoted as $\{e_{u,1}^k, e_{u,2}^k, \ldots, e_{u,|A|}^k\}$. By aggregating these embeddings in $|A|$ attribute domains, the ultimate user and item embeddings for the behavior k are detailed as below:

$$\begin{aligned} e_u^k &= \sum_{a \in A} e_{u,a}^k \\ e_i^k &= \sum_{a \in A} e_{i,a}^k. \end{aligned} \tag{12}$$

Utilizing the aforementioned function, we can derive the user and item embeddings for behaviors spanning $\{1, 2, \ldots, K\}$, represented as $\{e_u^1, e_u^2, \ldots, e_u^K\}$.

Denoised contrastive learning

As mentioned in the introduction, CL can effectively synchronize user embeddings in multiple behaviors, which motivates us to employ CL technique for the embeddings alignment across different behaviors. However, the user embeddings in auxiliary behaviors contain information that is less relevant to the user embedding in the target behavior, which refers to the noise data for the CL. The noise data will degrade the distinguishing ability for positive and negative samples in CL, thus can not align user preference in multiple behaviors effectively.

In response, we introduce the denoised contrastive learning (DCL), which conducts contrastive learning to focus more on the consistent parts between user preferences in auxiliary behaviors and target behavior, reducing the impact of irrelevant information that brings noise data. Specifically, we first introduce the attention mechanism to obtain attention scores of user embeddings for different attribute domains in auxiliary behaviors, these scores reflect the importance of different attribute domains in auxiliary behaviors towards the user preference in the target behavior. Then, the attention scores are used to aggregating user embeddings for different attribute domains, thus we can obtain the final denoised user embeddings in auxiliary behaviors. Finally, we conduct CL utilizing these denoised user embeddings in auxiliary behaviors to align user preferences in multiple behaviors precisely.

Multi-attributes linearized attention mechanism Inspired by the self attention designed by [70, 71], we adopt the multi-attributes linearized attention mechanism to implement efficient and simple attention mechanism. We view the user embeddings in the target behavior as Q, user embeddings in the auxiliary behavior as K and V, and we obtain Q, K and V as below:

$$\begin{aligned} Q &= W_Q \cdot E_u^K \\ K &= W_K \cdot E_u^k \\ V &= W_V \cdot E_u^k \end{aligned} \tag{13}$$

where $E_u^K \in \mathbb{R}^{|A| \times d}$ denotes user feature matrix of target behavior, $E_u^k \in \mathbb{R}^{|A| \times d}$ denotes user feature matrix of k_{th} auxiliary behavior, which is the concatenation of $e_{u,1}^k, e_{u,2}^k, \ldots, e_{u,|A|}^k$ learned in views of different item attribute domains. $|A|$ denotes the number of the attribute domains. $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$ denotes learned parameter matrices.

Then, we employ the linearized attention mechanism on the user feature of k_{th} auxiliary behavior for the attribute a, using the function as follows to compute the attentive embeddings:

$$\hat{E}_u^k = (\phi(Q)\phi(K^T))V \tag{14}$$

where $\phi()$ is the feature map function. Noted that linearized attention mechanism we adopt can effectively reduce model complexity. We adopt the function as below to measure the positive similarity:

$$\phi(x) = \text{elu}(x) + 1 \tag{15}$$

where $\hat{E}_u^k \in \mathbb{R}^{|A| \times d}$ is consisted of the slices of $[\hat{E}_{u,1}^k, \hat{E}_{u,2}^k, \ldots, \hat{E}_{u,|A|}^k]$, and $|A|$ denotes the number of attribute domains. The final user feature matrices of the auxiliary behavior $\hat{e}_u^k \in \mathbb{R}^d$ can be computed by adding slices of the matrix for the attribute dimension as below:

$$\hat{e}_u^k = \sum_{a \in A} \hat{E}_{u,a}^k. \tag{16}$$

Contrastive learning. Based on the attention mechanism, we can obtain the denoised user embeddings in auxiliary behaviors as $\hat{e}_u^1, \hat{e}_u^2, \ldots, \hat{e}_u^{K-1}$, and the user embeddings in the target behavior e_u^K. Ultimately, we utilize the CL to align user preferences in auxiliary behaviors and target behavior for a batch of users U. The contrastive learning loss designed follows the previous contrastive learning based recommendations such as [17, 72], and the final InfoNCE-based contrastive learning loss for the behavior k is designed as below:

$$\mathcal{L}_{cl\_k}^{\text{user}} = \sum_{u \in U} -\log \frac{\exp(\phi(\hat{e}_u^k, e_u^K)/\tau)}{\sum_{v \in U \backslash u} \exp(\phi(\hat{e}_v^k, e_u^K)/\tau)} \tag{17}$$

where k denotes the k_{th} auxiliary behavior, τ is the temperature hyper-parameter, which is set as 0.5 in this work, $\phi()$ denotes the inner-product of two vectors, $v \in U \backslash u$ denotes the users except the user u in the user set. The goal of the function is to minimize the difference of preference representations in the user's auxiliary behavior and the target behavior. Due to the embeddings used for CL are denoised for different attribute domains, the model can align the user preference between the auxiliary behavior and the target behavior in a fine-grained way. We obtain the final CL loss by summing all the CL losses of auxiliary behaviors. The final contrastive learning function is designed as below:

$$\mathcal{L}_{cl} = \sum_{k=1}^{K-1} \mathcal{L}_{cl\_k}^{\text{user}}. \tag{18}$$

Model optimization

For embeddings learned from DGCN, we concentrate user and item embeddings in multiple behaviors, then utilize the weight matrix to obtain user and item embeddings q_u^g and q_i as below:

$$\begin{aligned} q_u^g &= W_u([e_u^1; e_u^2; \ldots; e_u^K]) + b_u \\ q_i &= W_i([e_i^1; e_i^2; \ldots; e_i^K]) + b_i \end{aligned} \tag{19}$$

where $e_u^1, e_u^2, \ldots, e_u^K, e_i^1, e_i^2, \ldots, e_i^K$ denotes user and item embeddings in multiple behaviors learned by Formula 12, $W_u \in \mathbb{R}^{(K \times d) \times d}$ and $W_i \in \mathbb{R}^{(K \times d) \times d}$ are weight matrices for the learned hyper-parameters, $b_u \in \mathbb{R}^d$, $b_i \in \mathbb{R}^d$ are bias parameters.

In addition, we further take advantage of the denoised user embeddings learned in DCL to represent user preference in auxiliary behaviors, and we adopt the same strategy to obtain denoised user embeddings q_u^d as below:

$$q_u^d = W_u^{'}([\hat{e}_u^1; \hat{e}_u^2; \ldots; \hat{e}_u^{K-1}; e_u^K]) + b_u^{'} \tag{20}$$

where $\hat{e}_u^1, \hat{e}_u^2, \ldots, e_u^K, \ldots, \hat{e}_u^{K-1}$ denotes user embeddings in auxiliary behaviors learned by Formula 16, $W_u^{'} \in \mathbb{R}^{(K \times d) \times d}$ is weight matrix, $b_u^{'} \in \mathbb{R}^d$ is bias parameter.

We incorporate the two user representations mentioned above to determine the ultimate user preference. The combination of these two features can achieve denoised learning for different attribute domains at both item-level and behavior-level. Consequently, it leads to a more precise and enriched understanding of user preferences. The final user embedding q_u is represented as below:

$$q_u = q_u^g + q_u^d. \tag{21}$$

Afterwards, we optimize the model based on the user history records in the target behavior (e.g. purchase behavior). We utilize the BPR loss [31] for the model optimization, which is denoted as below:

$$\mathcal{L}_{bpr} = \sum_{(u,i,j) \in O} \log\{\sigma(q_u^T q_i - q_u^T q_j)\}, \tag{22}$$

where O denotes the training sample set, u denotes the user u, i denotes the positive sampled item i, j denotes the negative sampled item j, T means the transposition operation of vectors. q_u is the embedding of the user u, q_i is the embedding of the positive item i, q_j is the embedding of the negative item j. The goal of the equation is to maximize the differences of user preferences towards positive items and negative items. Ultimately, our model is optimized by the joint function as below:

$$\mathcal{L} = \mathcal{L}_{bpr} + \alpha \mathcal{L}_{cl} + \beta \mathcal{L}_{ind} + \mu \| \Theta \|_2^2 \tag{23}$$

where α, β and μ are hyper-parameters to control the proportion of contrastive learning loss, dependency loss and ℓ_2 regularization, respectively.

The complete process of our model is detailed in Algorithm 1, the model's output provides the score which denotes the likelihood that a user will interact with an item under the target behavior. Subsequently, we rank the candidate items according to their scores, ultimately selecting the Top-N items for recommendations.

Complexity analysis

The computational expense of our model is primarily attributed to the GCN and the multi-attributes linearized attention mechanism. As for the GCN. The quantity of edges of the graph is represented as $|E|$, the variety of behavior types is indicated as $|B|$, the number of the attribute domains is $|A|$. We also denote M and N to indicate the number of users and items respectively. The number of GCN layers L and the embedding size D are also needed to compute the computational expense. The computational expense of constructing the adjacency matrix is $O(2|E||A||B|)$, the computational expense of the target-guided behavior weight is $O(2|E||A||B|NDL)$, the computational expense of the graph convolution is $2|E||A||B|LD$, thus the computational expense of the GCN in our model is $O(2|E||B||A| + 2|E||B||A|NDL + 2|E||A||B|LD)$. The computational expense of the linearized attention mechanism in our model is $O(M|A|D^2)$ [71]. Therefore, the total computational expense of our model is approximately $O(2|E||A||B| + 2|E||A||B|NDL + 2|E||A||B|LD + M|A|D^2)$. Comparing to other GCN-based methods, our model can achieve competitive performance

Algorithm 1 Algorithm of DMR model

```
Require: Interaction matrices $Y_1$, $Y_2$,..., $Y_K$ under multiple types of behaviors, the number of attribute domains $|A|$, number of layers of GCN $L$, user $\mathcal{U}$ and item set $\mathcal{I}$.
Ensure: The estimated score of user $u \in \mathcal{U}$ towards item $i \in \mathcal{I}$
1: Initialize user embedding $e_u$ and item embedding $e_i$ for $u \in \mathcal{U}$ and $i \in \mathcal{I}$.
2: for $epoch \leftarrow 0, 1, 2...$ do
3:     for $step \leftarrow 0, 1, 2...$ do
4:         //Generate disentangled user and item embeddings
5:         for $k \leftarrow 0, 1, 2..., K$ do
6:             Get user $u$ and item $i$ embeddings for behavior $k$: $e_u^k = e_u, i_u^k = i_u$
7:             for $a \leftarrow 0, 1, 2..., |A|$ do
8:                 Get user and item embeddings for attribute domain $a$ by Eqs. 3–8
9:                 Calculate the dependency loss by Eqs. 4–6
10:            end for
11:        end for
12:        //Disentangled graph convolutional network
13:        for $k \leftarrow 0, 1, 2..., K$ do
14:            for $a \leftarrow 0, 1, 2..., |A|$ do
15:                for $u \in \mathcal{U}$ do
16:                    for $i \in \mathcal{I}$ do
17:                        Get target behavior guided weight by Eqs. 9, 10
18:                    end for
19:                end for
20:                for $u \in \mathcal{U}$ do
21:                    for $l \in L$ do
22:                        Get user embedding $e_{u,a}^{k,l+1}$ by Eq. 11
23:                    end for
24:                end for
25:                for $i \in \mathcal{I}$ do
26:                    for $l \in L$ do
27:                        Get item embedding $e_{i,a}^{k,l+1}$ by Eq. 11
28:                    end for
29:                end for
30:                //Denoised Contrastive Learning
31:                Get the user embeddings under auxiliary behavior by Eqs. 13–15
32:                Aggregate user embeddings under auxiliary behavior by Eq. 16
33:                Calculate the contrastive learning loss by Eqs. 17, 18
34:            end for
35:            Aggregate user embedding $e_u^k$ by Eq. 12
36:            Aggregate item embedding $e_i^k$ by Eq. 12
37:        end for
38:        Get ultimate user and item embeddings by Eqs. 19, 20
39:        Calculate the BPR loss by Eq. 22
40:        Get the ultimate loss by Eq. 23
41:    end for
42: end for
```

without much sacrifice on the computational complexity. In addition, the attribute domain is a latent semantic representation that is not affected by the number of product attributes in the real recommendation scenarios, and the number of attribute domains is limited through experimental analysis. Therefore, our model achieves a computational complexity similar to most models and is suitable for most recommendation scenarios.

Experiment

We conduct comprehensive experiments on three publicly available datasets to estimate the efficacy of our model. Specifically, we endeavor to address the following research inquiries:

RQ1: How effective is our model in MBR task compared to existing advanced methods?

RQ2: What is the significance of each module in our model towards the recommendation effectiveness?

RQ3: What influences do the individual hyper-parameters have on the recommendation performance?

RQ4: How do multiple behaviors contribute to the efficacy of the recommendation?

RQ5: How does the model perform under different sparsity radios?

RQ6: How does the attention mechanism works in this paper?

RQ7: How does the model perform with different strategies of the contrastive learning?

Experimental settings

Dataset

We adopt two real-world datasets in the MBR domain: Retailrocket and Beibei. These two datasets are already publicly available datasets.

1. *Retailrocket.* This benchmark dataset is obtained from the Retailrocket platform,[1] including user activities such as Page View, Add-to-Cart, and Transaction. We adopt the leave-one strategy to deal with this dataset. For each user, we select one user-item interaction for validation and another one user-item interaction for testing.
2. *Beibei.* This dataset is collected from Beibei[2] that is the primary online retail platform for baby products in China, we adopt the dataset published at. [3] This dataset mainly contains View, Cart, and Purchase behaviors. The dataset has been already divided into train and test parts, and we use the leave one strategy to select one user-item interaction record from the train set for each user to compose the validation set.

The statistics of the two datasets are detailed in Table 2. In two datasets, transaction and purchase behavior are viewed as the target behavior, with other forms of interactions being considered as auxiliary behaviors. For duplicate data in the

[1] https://tianchi.aliyun.com/dataset/146321.

[2] https://www.beibei.com/.

[3] https://github.com/MC-CV/CIGF/tree/main/Datasets/beibei.

Table 2 The statistics of datasets

Dataset	#Users	#Items	Behaviors
Retailrocket	2174	30113	{View,Cart,Transaction}
Beibei	21716	7977	{View,Cart,Purchase}

training set, we only retain the first occurrence of user-item interactions.

Baselines

We compare our model with numerous state-of-the-art recommendation models. The baseline can be divided into two categories: single-behavior models and multi-behavior models. We provide a succinct overview of these methods as follows.

Single-behavior based models:

- *BPR* [31]. It is a pairwise learning framework for implicit feedback recommendation, which is widely used for the item recommendation.
- *NCF* [4]. It is a neural framework for implicit feedback recommendation, which utilizes deep neural network to model collaborate information in user-item interactions.
- *LightGCN* [50]. It is a GCN based method, which simply adopts the linear aggregations to obtain the ultimate node embeddings, also removes the no-linear activation part of the typical GCN methods.
- *SGL* [72]. It is a method that combines contrastive learning and graph representation learning, achieving data augmentation through three methods include node dropout, edge dropout, and random walk, which improves the performance of graph representation learning through contrastive learning effectively.

Multi-behavior based models:

- *MBGMN* [12]: It employs a graph meta network to derive user and item feature representations by leveraging a unified heterogeneous user-item interaction graph, which enables the capture of comprehensive information of users and items, taking into account diverse behavioral information.
- *S-MBRec* [17]: It adopts GCN to derive user and item embeddings from numerous dependent user-item graphs, it also designs a star-style contrastive learning to align user preferences for different behaviors.
- *CIGF* [19]: It proposes a framework founded on a compressed interaction graph. This model can effectively addresses the challenges of imbalanced data and the sparse distribution of the target behavior.

Evaluation metrics

We utilize recall and NDCG to evaluate the performance of our model and all the baselines for each user, and the final results of two metrics are the average values over all users.

- *Recall@N:* It measures how many relevant instances are recommended within the Top-N results. This metric focuses on whether the items that users are interested in can be captured by the model.

$$Recall@N = \frac{|S(N;u)|}{T(u)} \tag{24}$$

  where $S(N;u)$ denotes the number of items appear in the recommended list (N items) that have already been interacted with the user u, $T(u)$ denotes the items that have already been interacted with the user u.
- *NDCG@N:* It assigns higher scores to hits that appear at elevated positions within the ranking list. This metric focuses on the position of the items in the recommended list that users are interested in.

$$\begin{aligned} DCG@N &= \sum_{i=1}^{K} \frac{2^{1\{u(i)=1\}-1}}{\log(i+1)} \\ NDCG@N &= \frac{1}{Z} DCG@N \end{aligned} \tag{25}$$

  where $1\{\}$ indicates the indicator function, $u(i) = 1$ denotes the function value is 1 if user u interacts with the item i, N denotes the number of items recommended, Z denotes normalization parameter.

Parameter settings

In our experiments, the grid search is introduced to tune different hyper-parameters, including learning rate among $[1e^{-1}, 1e^{-2}, 1e^{-3}, 1e^{-4}]$, the regularization weight among $[1e^{-1}, 1e^{-2}, 1e^{-3}]$, the number of attribute domains in the range of $[2, 3, 4, 5, 6, 7, 8, 9, 10]$, CL parameters from $[1e^{4}, 1e^{-3}, 1e^{-2}, 1e^{-1}, 1, 10]$, and the number of the GCN layer in the range of $[1, 2, 3]$. In addition, we fix the the batch-size and embedding size as 1024 and 32 for all the models, and fix the seed as 2023. Other hyper-parameters within the baselines are fine-tuned following the their original publications. The epoch number of the model running is set as 20 and we adopt the early-stopping strategy to avoid the over-fitting problem. We train our model on the train set, then valid the model on the validation set, save the best-performing parameters and obtain the model accuracies on the test set. On the Retailrocket dataset, the learning rate is set as $1e^{-3}$, the regularization weight is set as $1e^{-3}$, the number of the GCN layer is set as 2, the number of attribute domains is set as 3,

Table 3 Size of trainable parameters for single-behavior models

Method	BPR	NCF	LightGCN	SGL
Retailrocket	516592	517136	516592	1033184
Beibei	475088	475633	475088	950176

Table 4 Size of trainable parameters for multi-behavior models

Method	MBGMN	SMBRec	CIGF	DMR
Retailrocket	840192	3171308	1027048	3120486
Beibei	798688	2922284	854040	2871462

the CL parameter is set as $1e^{-3}$. On the Beibei dataset, the learning rate is set as $1e^{-2}$, the regularization weight is set as $1e^{-3}$, the number of the GCN layer is set as 2, the number of attribute domains is set as 3, the CL parameter is set as $1e^{-1}$. The comparison of the number of trainable parameters is shown in Tables 3, 4. Next, we summarize and analyze the experimental results on the test set.

Overall performance (RQ1)

To answer RQ1, we first perform experiments to compare the performance of our model DMR with all baselines, and the overall performance comparison results on two datasets are presented in Tables 5, 6, 7, and 8. The results produced by the best baseline and the best performer in each column are underlined and boldfaced, respectively. The last line represents the improvements proportions of our model compared to the best baseline. The results produced by the best baseline and the best performer in each column are underlined and boldfaced, respectively. These are the inferences we can draw from the results:

Overall, DMR achieves the best performances on two datasets. It can be seen that DMR improves on average 3.12% on the Retailrocket dataset and 3.28% on the Beibei dataset over the second best baseline, specifically improves approximately on average 3.75% and 2.49% respectively for Recall and NDCG metrics on Retailrocket dataset, 3.76% and 2.80% for Recall and NDCG metrics on the Beibei dataset. It can be seen that our model is stronger than current advanced research methods generally with obvious improvements on two metrics. On the Retailrocket dataset, our model exhibits remarkable enhancements for Recall@10 and Recall@20 metrics with 4.45% and 7.74%, respectively. On the Beibei dataset, it also shows significant improvements for Recall@10 and NDCG@40 metrics with 6.89% and 6.91%. These results demonstrate the effectiveness of our model in augmenting the performance of multi-behavior recommendations especially with limited numbers of recommendation list length. In addition, we also notice that when the length of the recommendation list is set as 80, the performance improvements of our model is not significant as other conditions. This may be because when the length of the recommendation list is too long, leading to the probability of random hits is higher, and the ability of the model itself to improve performance is no longer as significant. In addition, the metrics in our work for all models are at the range of 1–5% on two datasets, The range of metrics are determined by the characteristics of the dataset. When calculating metrics, negative samples refer to all items in the dataset, so the number of candidate items to some extent determines the range of metrics. And this result range is consistent with the results obtained by other methods using these two datasets. By comparing the performance improvements of the model with baselines, we can verify the effectiveness of the model.

Among all the single-behavior recommendation models, LightGCN performs much better than BPR and NCF methods. This demonstrates the crucial role of higher-order neighbors' information in enhancing the efficacy of user and item embedding learning for recommendations. It can also be seen that LightGCN and SGL achieve similar performance on two datasets, which demonstrates GCN and CL can improve the recommendation performance effectively. We can observe that all the multi-behavior baselines in our experiment achieve superior performances than all the single-behavior models on two datasets, which demonstrates the potential of multi-behavior information for enhancing recommendation accuracy.

Among all the multi-behavior recommendation models, MBGMN and CIGF all utilize graph neural network, while CIGF is superior over MBGMN on two datasets, which is probably due to CIGF can capture rich relationships between different behaviors by modeling high-order GCN, instead of the typical GCN adopted by MBGMN. CIGF achieves best performances in most cases, it improves the second best baseline S-MBRec by 2.22% and 6.39% for Recall and NDCG metrics on two datasets on average, and achieves 4.31% improvement on average. Our method achieves similar improvements with CIGF, but the performance of CIFG is not always better than S-MBRec in all cases. Comparing with it, our method achieves significant improvements in all cases, which also proves that our mode's ability for improving the MBR models. We can also observe that S-MBRec performs better than MBGMN on the Beibei dataset for all metrics, despite MBGMN achieving superior Recall@20, Recall@40, NDCG@10, and NDCG@20 results on Retailrocket dataset, S-MBRec improves 1.6% than MBGMN on average. This reveals that contrastive learning is instrumental in optimizing the performance of MBR models, which is because CL can help to capture similar preferences across different user behaviors and reduce noise data in auxiliary behaviors.

In conclusion, the performance evaluations conducted by all the implemented methods yield the following insights: (1)

Table 5 Overall performance comparisons on Retailrocket dataset for recall metric

Metric	Recall@10	Recall@20	Recall@40	Recall@80	Average
Single-behavior					
BPR	0.0225	0.0299	0.0363	0.0428	0.0329
NCF	0.0244	0.0359	0.0391	0.0543	0.0384
LightGCN	0.0271	0.0290	0.0336	0.0400	0.0324
SGL	0.0258	0.0290	0.0373	0.0451	0.0343
Multi-behavior					
MBGMN	0.0400	0.0465	0.0538	0.0667	0.0518
S-MBRec	0.0405	0.0455	<u>0.0570</u>	0.0722	0.0538
CIGF	<u>0.0409</u>	<u>0.0465</u>	0.0567	<u>0.0741</u>	0.0546
DMR	**0.0423**	**0.0501**	**0.0580**	**0.0745**	0.0562
Imp(%)	4.45	7.74	2.29	0.53	3.75

Table 6 Overall performance comparisons on Retailrocket dataset for NDCG metric

Metric	NDCG@10	NDCG@20	NDCG@40	NDCG@80	Average
Single-behavior					
BPR	0.0140	0.0159	0.0172	0.0183	0.0164
NCF	0.0123	0.0154	0.0160	0.0191	0.0202
LightGCN	0.0162	0.0166	0.0176	0.0187	0.0173
SGL	0.0154	0.0162	0.0178	0.0191	0.0171
Multi-behavior					
MBGMN	0.0222	0.0239	0.0254	0.0276	0.0248
S-MBRec	0.0208	0.0238	0.0258	<u>0.0289</u>	0.0248
CIGF	<u>0.0228</u>	<u>0.0255</u>	<u>0.0263</u>	0.0286	0.0258
DMR	**0.0238**	**0.0261**	**0.0268**	**0.0292**	0.0264
Imp(%)	2.63	4.39	1.90	1.03	2.48

Table 7 Overall performance comparisons on Beibei dataset for Recall metric

Metric	Recall@10	Recall@20	Recall@40	Recall@80	Average
Single-behavior					
BPR	0.0172	0.0223	0.0585	0.1229	0.0552
NCF	0.0263	0.0423	0.0745	0.1216	0.0662
LightGCN	0.0291	0.0437	0.0647	0.0916	0.0572
SGL	0.0303	0.0395	0.0529	0.0668	0.0474
Multi-behavior					
MBGMN	0.0400	0.0633	0.0965	0.1514	0.0878
S-MBRec	0.0412	0.0643	0.0981	0.1515	0.0888
CIGF	<u>0.0421</u>	<u>0.0689</u>	<u>0.1097</u>	<u>0.1785</u>	0.0998
DMR	**0.0450**	**0.0704**	**0.1125**	**0.1846**	0.1031
Imp(%)	6.89	2.17	2.55	3.42	3.76

multi-behavior based models are superior to single-behavior based models by extracting more useful information from multiple user behaviors. (2) Modeling multi-behavioral interactions using GCN is an effective approach, (3) contrastive learning between multiple behaviors can enhance the performance of MBR trough consistency learning among multiple behaviors.

Ablation study (RQ2)

To answer RQ2, we evaluate the efficacy of each designed module incorporated into our model by examining three distinct variants:

Table 8 Overall performance comparisons on Beibei dataset for NDCG metric

Metric	NDCG@10	NDCG@20	NDCG@40	NDCG@80	Average
Single-behavior					
BPR	0.0062	0.0075	0.0148	0.0256	0.0135
NCF	0.0115	0.0155	0.0219	0.0300	0.0197
LightGCN	0.0153	0.0189	0.0232	0.0278	0.0213
SGL	0.0174	0.0198	0.0225	0.0249	0.0212
Multi-behavior					
MBGMN	0.0228	0.0292	0.0314	0.0407	0.0310
S-MBRec	<u>0.0238</u>	0.0295	0.0347	0.0458	0.0335
CIGF	0.0228	<u>0.0303</u>	<u>0.0362</u>	<u>0.0501</u>	0.0349
DMR	**0.0243**	**0.0309**	**0.0387**	**0.0502**	0.0360
Imp(%)	2.10	1.98	6.91	0.19	2.80

- *w/o target behavior-guided weight (TW):* We remove the weight in the GCN in our model, replacing our designed graph convolutional module with LightGCN method.
- *w/o multi-attributes linearized attention mechanism (MLAM):* We remove the attention mechanism in the contrastive learning, replace the denoised contrastive learning with universal contrastive learning.
- *w/o CL loss:* We remove the contrastive learning loss of our model.

The results of the ablation study are presented in Tables 9, 10, 11, and 12. Comprehensively, the entire model leads to the best performance on two datasets, illustrating that the integration of three modules significantly advances performances of MBR models. In addition, we found that removing TW or MLAM all resulted in a decrease in the model performance, indicating that the denoising learning of these two modules are useful for enhancing the model performance. Next, we will analyze the contributions of each component as follows:

On the Beibei dataset, There's a substantial decline in model efficacy for Recall@20 and Recall@40 metrics when we removed TW, and the full model improves 1.3% than the model without TW on average for all metrics. On the Retailrocket dataset, the model without TW achieves superior performance for Recall@20, while still performs worse than the full model in most cases. Results on both two datasets demonstrate the advantageous of the TW module we designed. As for the situation that the model without TW module performs better than the full model, it is probably due to over-smoothing problem for the weighted GCN. In addition, the model without TW still achieves superior performance than MBGMN and SMBRec, which demonstrates the combination of the CL and MLAM is also important for improving the model performance.

Additionally, the full model improves 6.78% and 3.47% on Beibei and Retailrocket datasets respectively than the model without CL, 6.03% and 2.93% than the model without MLAM. This observations indicate that CL and MLAM module both play an essential roles on the two datasets. Moreover, the absence of MLAM in the model results in a more significant decline in performance than the situation that CL is omitted, which is probably due to the limitations of traditional CL when confronted with noise data. This result demonstrates the denoised part in the CL is functional, and the MLAM module we developed plays an essential role in enhancing the ability of the CL in our model. When the model removes TW, it still achieves superior performance, which further indicate the model's performance when CL and MLAM are combined. This combination is observed to chiefly enhance the NDCG metrics on two datasets, highlighting that denoised contrastive learning contributes to discovering user interests while also ranking the items user is interested at the top of the recommendation list.

Hyperparameter study (RQ3)

To answer RQ3, we conduct extensive experiments to investigate the effect of different hyper-parameters, which include the number of the GCN layer L, the number of the attribute domain $|A|$, and the balance parameters α of the CL. In the experiment, the random seeds are not fixed, and the maximum, minimum, and average values of the statistical results are obtained through several experiments. The error of the results is represented by the shaded area in Figs. 3, 4 and 5. Overall, the model performance is characterized by a small margin of error and high stability, which demonstrates the robustness of our model. The following are the results and analysis of the impact of different parameters on model performance.

Effect of the number of layers The Recall@10 and NDCG@10 results of all models on two datasets varying different values of L are shown in Fig. 3, where the number of attribute domains is set as 3 for two datasets, and the value of

Table 9 Ablation study performance comparisons on Retailrocket dataset for Recall metric

Method	Recall@10	Recall@20	Recall@40	Recall@80
w/o TW	**0.0426**	0.0478	0.0515	0.0713
w/o CL	0.0419	0.0460	0.0561	0.0704
w/o MLAM	0.0419	0.0465	0.0557	0.0708
Full model	0.0423	**0.0501**	**0.0580**	**0.0745**

Table 10 Ablation study performance comparisons on Retailrocket dataset for NDCG metric

Method	NDCG@10	NDCG@20	NDCG@40	NDCG@80
w/o TW	0.0238	0.0252	0.0260	0.0276
w/o CL	0.0235	0.0256	0.0267	0.0285
w/o MLAM	0.0237	0.0255	0.0262	0.0291
Full model	**0.0238**	**0.0261**	**0.0268**	**0.0292**

Table 11 Ablation study performance comparisons on Beibei dataset for Recall metric

Method	Recall@10	Recall@20	Recall@40	Recall@80
w/o TW	0.0423	0.0611	**0.1177**	**0.1973**
w/o CL	0.0430	0.0672	0.1072	0.1799
w/o MLAM	0.0420	0.0667	0.1128	0.1817
Full model	**0.0450**	**0.0704**	0.1125	0.1846

α is set as $1e^{-3}$ and $1e^{-1}$ on Retailrocket and Beibei dataset. Overall, our model outperforms other baselines with different L values in most cases, the performance of the proposed model is only slightly lower than that of the CIGF model when the value of L is set as 3 and 5. We also found that the performance changes of all models are more stable on the Beibei dataset than on the Retailrocket dataset, which is probably due to the Retailrocket dataset being sparser compared to the Beibei dataset, and the larger values of L on the sparser dataset would cause over-smoothing problems. In addition, almost all the models achieve the best performance when the value of L is set as 2 on two datasets, and there is a decreasing trend when the value of L is larger than 2. This is because the over-smoothing problem will lead to worse performance with the larger values of L. Among all the baselines, the decreasing trend of the performance of the CIGF model is more obvious compared to other models, which is probably due to the CIGF model relies entirely on a complex GCN, making the consequences of differing numbers of GCN layers on model effectiveness are more apparent.

Effect of the number of the attribute domain The Recall@10 and NDCG@10 results of all the models on two datasets varying different numbers of the attribute domain are shown in Fig. 4, where the number of the value of L is set as 2 on two datasets, and the value of α is set as $1e^{-3}$ and $1e^{-1}$ on Retailrocket and Beibei datasets respectively. The results indicate there is a declining trend in model performance as the number of attribute domain increases from 2 to 5, the model achieves the worst performance when the number of the attribute domain is set as 5. Then, there is an increasing trend of the model performance on two datasets when the number of attribute domain is larger than 5. Further observation reveals a rising trend in the model's efficacy as the value of $|A|$ increases from 2 to 3, which demonstrates that an elevation in the number of attribute domains enables the model to obtain more precise information comprehension. However, more noise data will be introduced with the increasing number of attribute domains, which may be the reason for the decrease in the model performance with the value of $|A|$ increasing from 3 to 5. With the increase of the value of $|A|$, more useful information will be introduced, which is

Table 12 Ablation study performance comparisons on Beibei dataset for NDCG metric

Method	Recall@10	Recall@20	Recall@40	Recall@80
w/o TW	0.0252	0.0297	0.0385	**0.0520**
w/o CL	0.0235	0.0296	0.0308	0.0459
w/o MLAM	**0.0237**	0.0300	0.0287	0.0516
Full model	0.0235	**0.0309**	**0.0387**	0.0502

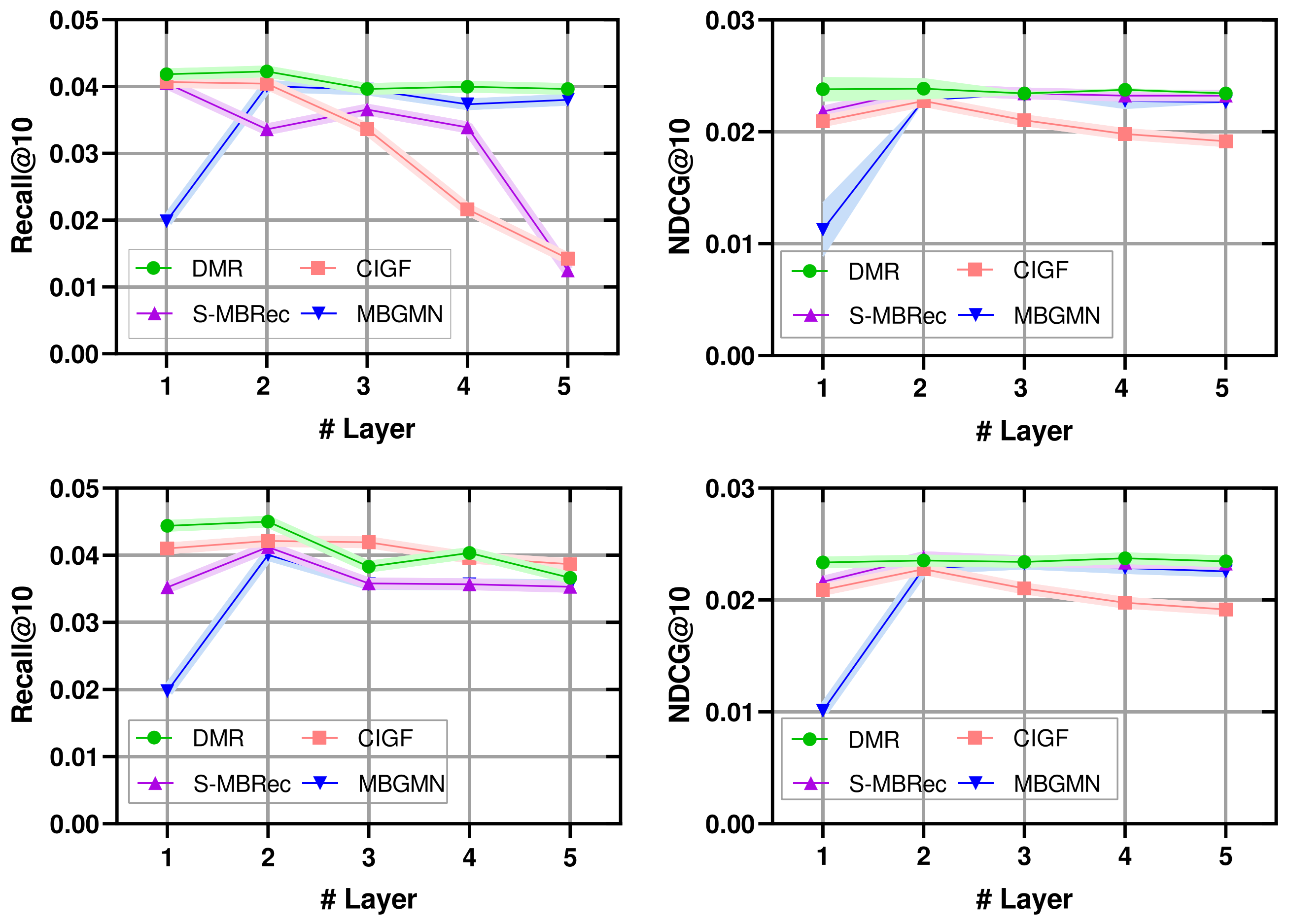

Fig. 3 Performance comparison with different numbers of layer. The above two figures show the model performance on the Retailrocket dataset, and the following two figures show the model performance on the Beibei dataset

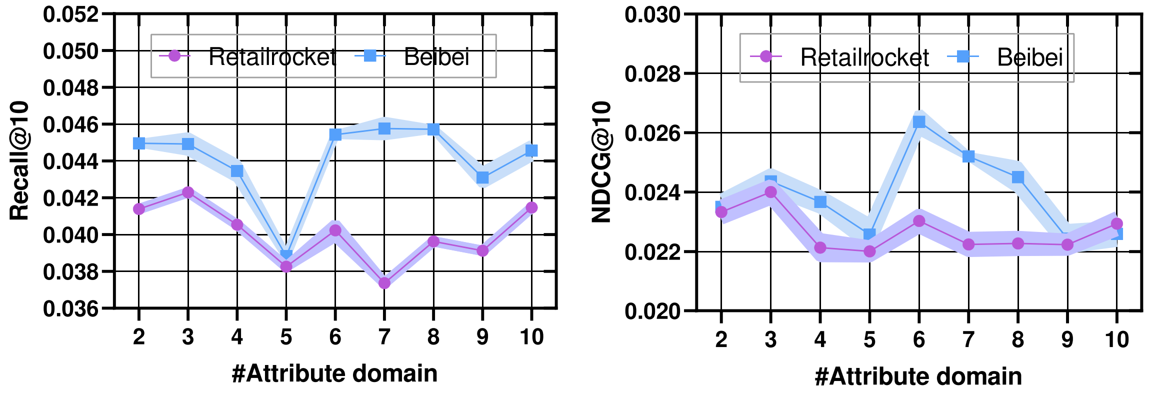

Fig. 4 Performance comparison with different numbers of attribute domains on two datasets

probably the reason the model performance becomes much better. In addition, we can observe that the model achieves similar performance when the value of $|A|$ is less than 5 and larger than 5, which is probably due to the ability of the dependent loss will be limited with the larger value of $|A|$, leading to the item embeddings are similar for different attribute domains, thus resulting in the model degradation. Based on the above discussion, it can be observed that the model achieves effective performance on both datasets when the number of attribute domains is set as 2 and 3. Therefore, there is no need to set the number of attribute domains too large, a small number of attribute domains will ensure model performance while reducing model complexity. Although the performance of the model fluctuates under different numbers

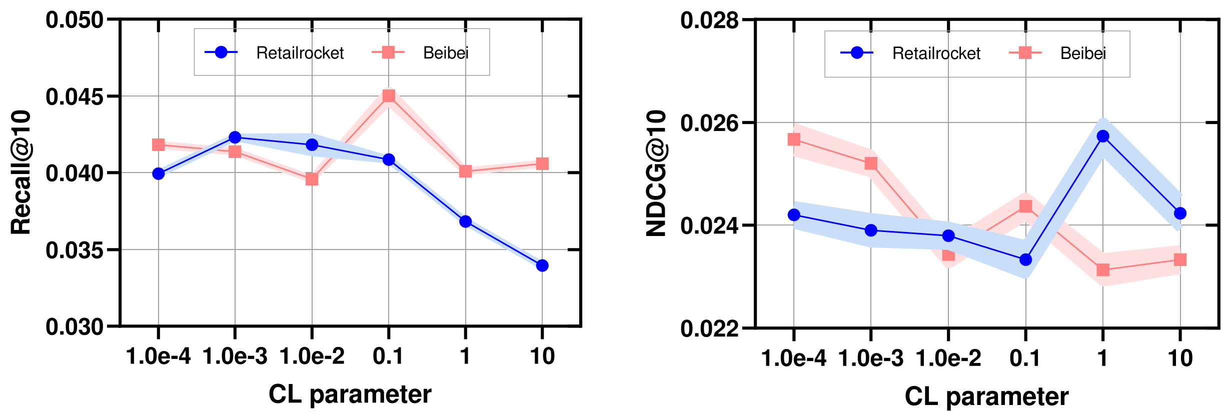

Fig. 5 Performance comparison with different values of α on two datasets

of attribute domains, it's overall performance is relatively stable, e.g. the Recall metric on Beibei dataset fluctuates around 0.044. In addition, the model performance is still higher than baselines under different numbers of attribute domains in most cases, which demonstrates modeling user preference in view of attribute domain is functional.

Effect of balance parameters The Recall@10 and NDCG@10 results of all the models on two datasets varying different values of α are shown in Fig. 5, where the number of the value of L is set as 2 on two datasets, and the value of $|A|$ is set as 3 on two datasets respectively. This results reveal that on the Retailrocket dataset, the best Recall outcomes are achieved by setting α as $1e^{-3}$, while the best NDCG outcomes are accomplished by setting α as 1. On the Beibei dataset, our model's performance for Recall metric is most optimal when the value of α equals $1e^{-1}$, and its performance for NDCG metric is best when the value of α is set as $1e^{-4}$. We found that the larger value of α is more beneficial for improving the NDCG result on the Retailrocket dataset, and is more beneficial for improving the recall result on the Beibei dataset. The explanation for these results can be attributed to the following analysis: (1) the recall results for different α values indicate the strength of the CL module for model's capacity, we can observe that the CL contributes more to the Recall metric on the Beibei dataset than the Retailrocket dataset. It is probably because there is a higher count of items than users on the Retailrocket dataset, leading to the weight of contrastive learning strategy built on user preferences is less significant on the Retailrocket dataset than Beibei dataset. Additionally, the Retailrocket dataset is sparser in comparison to the Beibei dataset, the Beibei dataset's higher density could lead to an increased presence of noise data, which consequently calls for increased contrastive learning weights to ensure proper denoised learning. (2) The NDCG results for different α values indicate the strength of the CL module for ranking the user's interest, we can observe that there is a decreasing trend of the NDCG results as the proportion of CL increases on Retailrocket dataset. The underlying cause of this phenomenon is likely the higher number of users than items leads to a more impact of the CL in reducing the noise data, whereas excessive CL will despite improving the recall results but limits the potential of the model for capturing the diverse user preference, thus compromising its effectiveness in differentiating items in the recommended list.

Behaviors study (RQ4)

To analyze the effect of different behaviors, we conduct experiments for different combinations of the behaviors, the performance results are shown in Fig. 6. Observing the results, it becomes evident that our model reaches peak performance when embracing all user behaviors, this is mainly due to more types of behaviors enable the model to precisely estimate user preference through alleviating the data sparsity. Additionally, the model's efficacy is heightened with the use of the view behavior as opposed to the cart behavior, which indicates that the view behavior is more impactful than the cart behavior in improving model performance. This is because there contains various user-item information in the view behavior, enabling the learning of diverse user preferences across numerous aspects and augmenting the model's capability in dealing with sparseness issues. When comparing model performance between two datasets, it is apparent that the performance advancement through the incorporation of multiple behaviors is more significant on Beibei dataset than Retailrocket dataset, especially for the Top@80 recommendations. This is likely due to the Beibei dataset contains more denser user-item interactions in auxiliary behaviors, thus enabling the model to capture user preference utilizing more abundant auxiliary information.

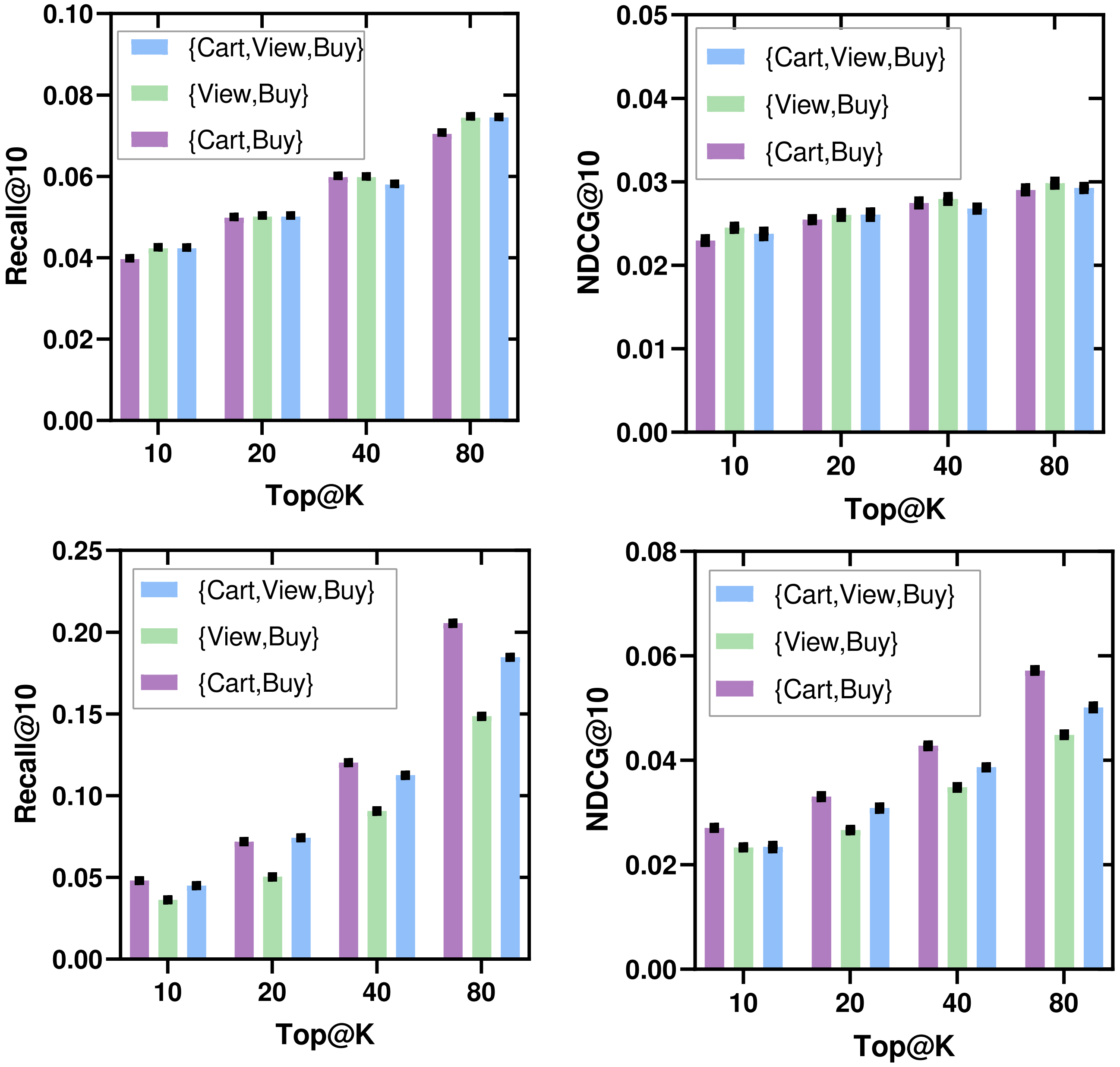

Fig. 6 Contributions of different behaviors on two datasets. The above two figures show the model performance on the Retailrocket dataset, and the following two figures show the model performance on the Beibei dataset

Sparsity study (RQ5)

To investigate the proposed model performance in dealing with the sparsity problem, we conduct extensive experiments on Beibei dataset with different dataset sparsity ratios. Specifically, we removed 20%, 30%, 40%, 50%, 60%, 70%, 80% and 90% of the user-item interactions existing in the data to simulate sparse situations, then compared the performance of the DMR model and LightGCN model under different sparsity ratios. The results are shown in Fig. 7. From the results we can observe that there is a decrease trend of two model performances with the increase of sparsity ratios. However, the performance of our model is superior over LightGCN with different sparsity ratios, indicating that our model can effectively handle the sparsity problem than the simple model. It is worth noting that when the data sparsity is larger than 70%, the recall and NDCG metrics of two models decrease significantly. This is due to in extremely sparse situations, the model cannot be fully trained, which will lead to over-fitting problem. We can also observe the the recall and NDCG metrics of LightGCN decrease more obviously than our model, which indicates that our model has significant abilities in extremely sparse situations comparing with simple models.

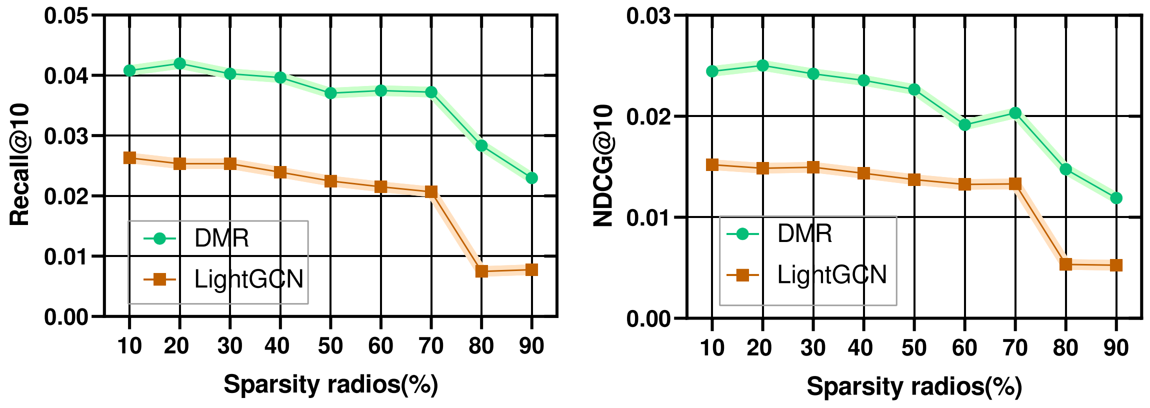

Fig. 7 Performance comparison with different sparsity radios

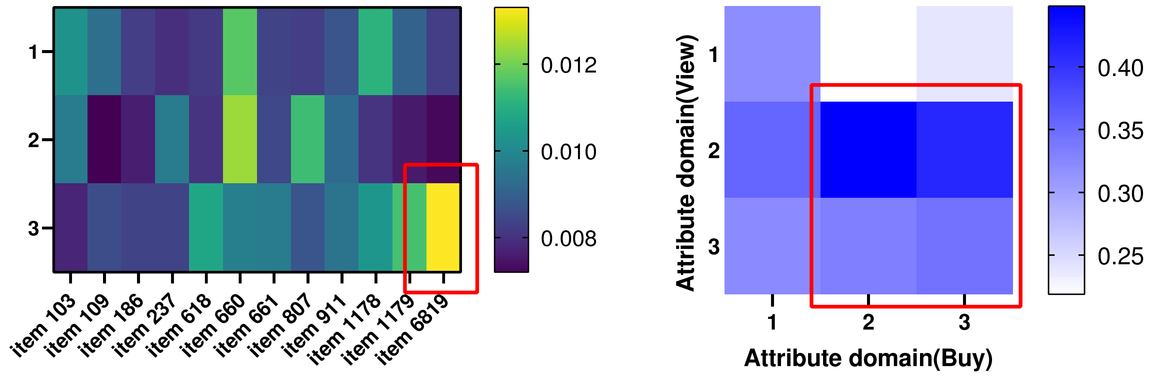

Fig. 8 Attention analyze. The left side represents the item weights in different attribute domains under view behavior. The right side represents the attention weights between view behavior and buy behavior for different attribute domains. The red boxes represent the parts with significant weights

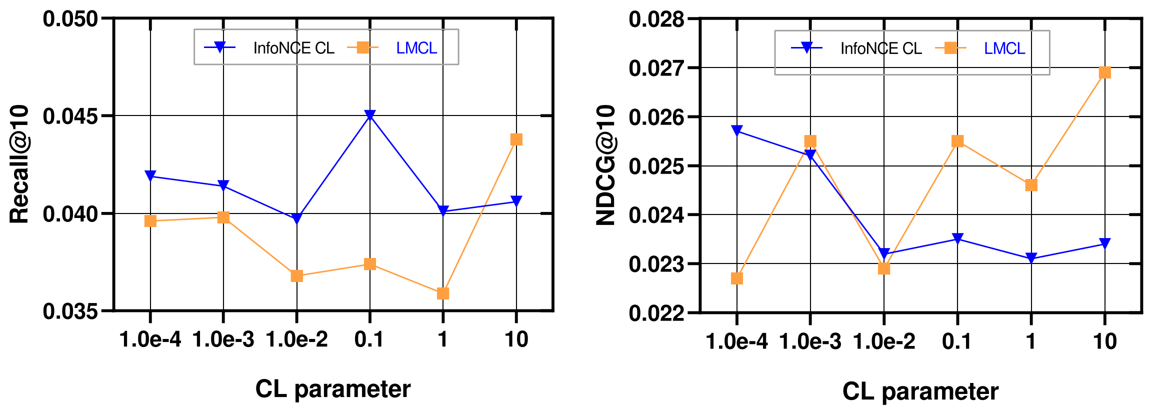

Fig. 9 Model performance on Beibei dataset for different contrastive learning strategies

Analysis of attention mechanism (RQ6)

To analyze the effect of the attention mechanism in CL and the target behavior-guided weighs in GCN. We conduct experiments on Beibei dataset and set the number of attribute domains as 3. We randomly select the user-item pairs (#20645, #6819) that are accurately predicted by the model in the test set, then we visualize the attention scores and graph aggregation weights for the user #20645, the results are shown in Fig. 8. The left part in the figure is the hot heat figure of the target behavior-guided weighs in view of attribute domain #3 under the view behavior. From the hot heat figure we can observe the predicted item has the highest weight during the propagation of GCN in the view of attribute domain #3, which demonstrates our model can effectively capture the item importance through target-guided module in view of the attribute domain. The right part in the figure shows the attention scores of three domains between the buy behavior and the view behavior. We can observe that the attention mechanism in CL achieves smaller weights of attribute domain #1 for the view behavior, while larger attention scores for both attribute domain #2 and #3. This demonstrates domain #2 and #3 are more important than domain #1 in the view behavior, thus the attention mechanism can effectively reduce the influence of the noisy attribute domains, which contributes to the denoising learning in CL. In addition, the model can ultimately learn higher weights for the item #6819 through two modules, demonstrating both two modules are essential for superior model performances.

Analysis of contrastive learning strategy (RQ7)

To explore the sensitivity of our model towards different CL strategies, we conduct experiments on Beibei dataset with two CL strategies varying different CL parameters α. We adopt large margin contrastive learning loss (LMCL) and InfoNCE contrastive learning loss (InfoNCE CL) respectively for the Formula 17, the main idea of large margin contrastive learning loss is to minimize the distance between samples that belong to the same class, while maximizing the distance between samples belonging to different categories, we utilize ℓ_2 to evaluate the distance between user embeddings in two behaviors. The results for Recall@10 and NDCG@10 are shown in Fig. 9. From the results we can observe that for Recall metric, InfoNCE CL is more effective than LMCL for improving the model performance, model with LMCL achieves best Recall values when the CL parameter is set as 10, but still slightly lower than model with InfoNCE CL. For NDCG metric, model with LMCL achieves best NDCG value when the CL parameter is set as 10, which is superior to model with InfoNCE CL. Based on the above discussions, we conclude that regardless of which contrastive loss function is used, our model can achieve significant performances.

Conclusion

In this paper, we propose a method that integrates disentangled convolutional network and denoised contrastive learning to address the MBR task. The methodology specifically starts with the generation of disentangled domains based on item attributes. Afterward, we learn user and item characteristics by modeling the user-item connections in each distinct attribute domain. Finally, we design the denoised contrastive learning (DCL) to more accurately harmonize user preferences across multiple behaviors. Experiment results on two datasets show our model improves on average 3.12% on the Retailrocket dataset and 3.28% on the Beibei dataset over the best baseline, which demonstrates the effectiveness of our model, and the further study results demonstrate the effectiveness of our model in fine-grained learning and denoised learning. In the future, we will conduct research on the data augmentation problem in the MBR task. We will simultaneously process noisy data from both the data and model perspectives, also considering the sequential information to deal with the noise issue more effectively.

Acknowledgements No conflict of interest exists in the submission of this manuscript. I would like to declare on behalf of my co-authors that the work described was original research that has not been published previously and is not under consideration for publication elsewhere, in whole or in part. The datasets can be obtained based on public websites, which has been indicated in the manuscript. Our codes will be released when the paper is accept. All the listed authors have contributed to the described work and approved the final manuscript.

Author Contributions Yijia Zhang: Methodology, Writing-Original Draft. Wanyu Chen: Conceptualization. Fei Cai: Writing-Review and Editing. Zhenkun Shi: Formal analysis. Feng Qi: Investigation.

Data Availability Statement All data associated with this study are publicly available and can be found at the URLs provided in the paper.

References

1. Jing M (2023) Contrastive self-supervised learning in recommender systems: a survey. ACM Trans Inf Syst 42(2):1–39
2. Li X, Xu S, Jin H, Wang Z, Ma Y, He X (2024) Poi recommendation by deep neural matrix factorization integrated attention-aware meta-paths. Complex Intell Syst 1–15
3. Mnih A, Salakhutdinov RR (2007) Probabilistic matrix factorization. In: Advances in neural information processing systems, vol 20
4. He X (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web, pp 173–182
5. Wang X, He X, Wang M, Feng F, Chua T-S (2019) Neural graph collaborative filtering. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 165–174
6. Guo H, Tang R, Ye Y, Li Z, He X (2017) Deepfm: a factorization-machine based neural network for ctr prediction. arXiv:1703.04247
7. Singh AP, Gordon GJ (2008) Relational learning via collective matrix factorization. In: Proceedings of the 14th ACM SIGKDD

international conference on knowledge discovery and data mining, pp 650–658
8. Chen Y, Cao Q, Huang X, Zou S (2024) Multi-behavior collaborative contrastive learning for sequential recommendation. Complex Intell Syst 1–16
9. Chen X, Li Z, Pan W, Ming Z (2023) A survey on multi-behavior sequential recommendation. arXiv:2308.15701
10. Shen Y, Ou B, Li R (2022) Mbn: towards multi-behavior sequence modeling for next basket recommendation. ACM Trans Knowl Discov Data (TKDD) 16(5):1–23
11. Jin B, Gao C, He X, Jin D, Li Y (2020) Multi-behavior recommendation with graph convolution networks. In: 43nd international ACM SIGIR conference on research and development in information retrieval
12. Xia L, Xu Y, Huang C, Dai P, Bo L (2021) Graph meta network for multi-behavior recommendation. In: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, pp 757–766
13. Qian Y, Zhao P, Li Z, Fang J, Zhao L, Sheng VS, Cui Z (2019) Interaction graph neural network for news recommendation. In: Web information systems engineering–WISE 2019: 20th international conference, Hong Kong, China, November 26–30, 2019, Proceedings 20. Springer, pp 599–614
14. Li X, Li R, Peng Q, Yao J (2024) Graph neural news recommendation based on multi-view representation learning. J Supercomput 1–19
15. Yu J, Xia X, Chen T, Cui L, Hung NQV, Yin H (2023) Xsimgcl: towards extremely simple graph contrastive learning for recommendation. IEEE Trans Knowl Data Eng 36(2):913–926
16. Zhang W, Mao J, Cao Y, Xu C (2020) Multiplex graph neural networks for multi-behavior recommendation. In: Proceedings of the 29th ACM international conference on information & knowledge management, pp 2313–2316
17. Gu S, Wang X, Shi C, Xiao D (2022) Self-supervised graph neural networks for multi-behavior recommendation. In: IJCAI, pp 2052–2058
18. Meng C, Zhai C, Yang Y, Zhang H, Li X (2023) Parallel knowledge enhancement based framework for multi-behavior recommendation. In: Proceedings of the 32nd ACM international conference on information and knowledge management, pp 1797–1806
19. Guo W, Meng C, Yuan E, He Z, Guo H, Zhang Y, Chen B, Hu Y, Tang R, Li X (2023) Compressed interaction graph based framework for multi-behavior recommendation. In: Proceedings of the ACM web conference 2023, pp 960–970
20. Liu H, Sun T, Zhang Z, Zheng H, Liu G, Yang Z, Wang X (2024) A novel multi-behavior contrastive learning and knowledge-enhanced framework for recommendation. In: International conference on intelligent computing. Springer, pp 399–410
21. Xuan H, Li B (2023) Temporal-aware multi-behavior contrastive recommendation. In: International conference on database systems for advanced applications. Springer, pp 269–285
22. Qiao Z, Yan H, Han L (2023) Mixmbr: contrastive learning for multi-behavior recommendation. In: International conference on database systems for advanced applications. Springer, pp 434–445
23. Chang J, Gao C, Zheng Y, Hui Y, Niu Y, Song Y, Jin D, Li Y (2021) Sequential recommendation with graph neural networks. In: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, pp 378–387
24. Yang K, Toni L (2018) Graph-based recommendation system. In: 2018 IEEE global conference on signal and information processing (GlobalSIP). IEEE, pp 798–802
25. Song W, Wang S, Wang Y, Liu K, Liu X, Yin M (2023) A counterfactual collaborative session-based recommender system. In: Proceedings of the ACM web conference 2023, pp 971–982
26. Guan X, Cheng Z, He X, Zhang Y, Zhu Z, Peng Q, Chua T-S (2019) Attentive aspect modeling for review-aware recommendation. ACM Trans Inf Syst (TOIS) 37(3):1–27
27. Hou Y, Yang N, Wu Y, Yu PS (2019) Explainable recommendation with fusion of aspect information. World Wide Web 22:221–240
28. Hou Y, Ouyang Y, Liu Z, Han F, Rong W, Xiong Z (2023) Multi-level and multi-interest user interest modeling for news recommendation. In: International conference on knowledge science, engineering and management. Springer, pp 200–212
29. Han Y, Wang H, Wang K, Wu L, Li Z, Guo W, Liu Y, Lian D, Chen E (2024) Efficient noise-decoupling for multi-behavior sequential recommendation. In: Proceedings of the ACM on web conference 2024, pp 3297–3306
30. Xiao J, Pan W, Ming Z (2024) A generic behavior-aware data augmentation framework for sequential recommendation. In: Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval, pp 1578–1588
31. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2012) Bpr: Bayesian personalized ranking from implicit feedback. arXiv:1205.2618
32. Loni B, Pagano R, Larson M, Hanjalic A (2016) Bayesian personalized ranking with multi-channel user feedback. In: Proceedings of the 10th ACM conference on recommender systems, pp 361–364
33. Chen C, Ma W, Zhang M, Wang Z, He X, Wang C, Liu Y, Ma S (2021) Graph heterogeneous multi-relational recommendation. In: Proceedings of the AAAI conference on artificial intelligence, vol 35, pp 3958–3966
34. Xia L, Huang C, Xu Y, Dai P, Bo L (2022) Multi-behavior graph neural networks for recommender system. IEEE Trans Neural Netw Learn Syst
35. Wu Y, Xie R, Zhu Y, Ao X, Chen X, Zhang X, Zhuang F, Lin L, He Q (2022) Multi-view multi-behavior contrastive learning in recommendation. In: International conference on database systems for advanced applications. Springer, pp 166–182
36. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Advances in neural information processing systems, vol 30
37. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
38. Yu C, Liu Y, Gao C, Shen C, Sang N (2020) Representative graph neural network. In: Computer vision–ECCV 2020: 16th European conference, Glasgow, August 23–28, 2020, Proceedings, Part VII 16. Springer, pp 379–396
39. Zhang M, Wu S, Yu X, Liu Q, Wang L (2022) Dynamic graph neural networks for sequential recommendation. IEEE Trans Knowl Data Eng 35(5):4741–4753
40. Wu S, Sun F, Zhang W, Xie X, Cui B (2022) Graph neural networks in recommender systems: a survey. ACM Comput Surv 55(5):1–37
41. Berg R, Kipf TN, Welling M (2017) Graph convolutional matrix completion
42. Wang X, Jin H, Zhang A, He X, Xu T, Chua T-S (2020) Disentangled graph collaborative filtering. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp 1001–1010
43. Wu L, Yang Y, Zhang K, Hong R, Fu Y, Wang M (2020) Joint item recommendation and attribute inference: an adaptive graph convolutional network approach. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp 679–688
44. Zhang J, Shi X, Zhao S, King I (2019) Star-gcn: stacked and reconstructed graph convolutional networks for recommender systems. arXiv:1905.13129
45. Wang X, He X, Cao Y, Liu M, Chua T-S (2019) Kgat: knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining, pp 950–958

46. Sun R, Cao X, Zhao Y, Wan J, Zhou K, Zhang F, Wang Z, Zheng K (2020) Multi-modal knowledge graphs for recommender systems. In: Proceedings of the 29th ACM international conference on information & knowledge management, pp 1405–1414
47. Sha X, Sun Z, Zhang J (2021) Hierarchical attentive knowledge graph embedding for personalized recommendation. Electron Commer Res Appl 48:101071
48. Wang Z, Wei W, Cong G, Li X-L, Mao X-L, Qiu M (2020) Global context enhanced graph neural networks for session-based recommendation. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp 169–178
49. Fan S, Zhu J, Han X, Shi C, Hu L, Ma B, Li Y (2019) Metapath-guided heterogeneous graph neural network for intent recommendation. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2478–2486
50. He X, Deng K, Wang X, Li Y, Zhang Y, Wang M (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, pp 639–648
51. Velickovic P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. Stat 1050(20):10–48550
52. Chen T, Kornblith S, Norouzi M, Hinton G (2020) A simple framework for contrastive learning of visual representations. In: International conference on machine learning. PMLR, pp 1597–1607
53. Gidaris S, Singh P, Komodakis N (2018) Unsupervised representation learning by predicting image rotations. arXiv:1803.07728
54. Hjelm RD, Fedorov A, Lavoie-Marchildon S, Grewal K, Bachman P, Trischler A, Bengio Y (2018) Learning deep representations by mutual information estimation and maximization. arXiv:1808.06670
55. Liu H, Cao Q, Huang X, Liu F, Zhang C, An J (2024) Multi-source information contrastive learning collaborative augmented conversational recommender systems. Complex Intell Syst 1–15
56. Nuo M, Han X, Zhang Y (2023) Contrastive learning-based music recommendation model. In: International conference on neural information processing. Springer, pp 370–382
57. Tutsoy O, Sumbul HE (2024) A novel deep machine learning algorithm with dimensionality and size reduction approaches for feature elimination: thyroid cancer diagnoses with randomly missing data. Brief Bioinform 25(4):344
58. Tutsoy O, Balikci K, Ozdil NF (2021) Unknown uncertainties in the covid-19 pandemic: multi-dimensional identification and mathematical modelling for the analysis and estimation of the casualties. Digit Signal Process 114:103058
59. Yu J, Yin H, Xia X, Chen T, Cui L, Nguyen QVH (2022) Are graph augmentations necessary? simple graph contrastive learning for recommendation. In: Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval, pp 1294–1303
60. Lin Z, Tian C, Hou Y, Zhao WX (2022) Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In: Proceedings of the ACM web conference 2022, pp 2320–2329
61. Xia L, Huang C, Xu Y, Zhao J, Yin D, Huang J (2022) Hypergraph contrastive collaborative filtering. In: Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval, pp 70–79
62. Cai X, Huang C, Xia L, Ren X (2023) Lightgcl: simple yet effective graph contrastive learning for recommendation. arXiv:2302.08191
63. Sang L, Hu Y, Zhang Y, Zhang Y (2024) Multi-view denoising contrastive learning for bundle recommendation. Appl Intell 54(23):12332–12346
64. Wang H, Xu Y, Yang C, Shi C, Li X, Guo N, Liu Z (2023) Knowledge-adaptive contrastive learning for recommendation. In: Proceedings of the sixteenth ACM international conference on web search and data mining, pp 535–543
65. Chen Y, Liu Z, Li J, McAuley J, Xiong C (2022) Intent contrastive learning for sequential recommendation. In: Proceedings of the ACM web conference 2022, pp 2172–2182
66. Yang Y, Huang C, Xia L, Huang C, Luo D, Lin K (2023) Debiased contrastive learning for sequential recommendation. In: Proceedings of the ACM web conference 2023, pp 1063–1073
67. Gao C, He X, Gan D, Chen X, Feng F, Li Y, Chua T-S, Yao L, Song Y, Jin D (2019) Learning to recommend with multiple cascading behaviors. IEEE Trans Knowl Data Eng 33(6):2588–2601
68. Cai W, Pan W, Mao J, Yu Z, Xu C (2022) Aspect re-distribution for learning better item embeddings in sequential recommendation. In: Proceedings of the 16th ACM conference on recommender systems, pp 49–58
69. Wang X, Huang T, Wang D, Yuan Y, Liu Z, He X, Chua T-S (2021) Learning intents behind interactions with knowledge graph for recommendation. In: Proceedings of the web conference 2021, pp 878–887
70. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, vol 30
71. Katharopoulos A, Vyas A, Pappas N, Fleuret F (2020) Transformers are rnns: fast autoregressive transformers with linear attention. In: International conference on machine learning. PMLR, pp 5156–5165
72. Wu J (2021) Self-supervised graph learning for recommendation. In: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, pp 726–735