Knowledge-Based Systems 338 (2026) 115449

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Integrating deep clustering and multi-view graph neural networks for recommender system

Jiaxuan Song[a,b,1], Yue Li[a,c,1], Duantengchuan Li[a,b,*], Xiaoguang Wang[a,b,*], Rui Zhang[a], Hui Zhang[a], Jinsong Chen[d]

[a] *School of Information Management, Wuhan University, 430072, Wuhan, China*
[b] *Intelligent Computing Laboratory for Cultural Heritage, Wuhan University, 430072, Wuhan, China*
[c] *Department of Intelligent Construction, School of Civil Engineering, Wuhan University, 430072, Wuhan, China*
[d] *Faculty of Artificial Intelligence Education, Central China Normal University, 430079, Wuhan, China*

A R T I C L E I N F O

Keywords:
Recommender system
Deep clustering
Contrastive learning
Collaborative filtering

A B S T R A C T

The existing graph neural network recommendation models aggregate neighborhood information by using a weighted sum strategy based on node popularity. However, this strategy struggles to accurately model the impact of item category features on user behavior. To alleviate this problem, we propose MDCRec, a novel graph convolutional recommendation framework integrating deep clustering. MDCRec utilizes the deep clustering module to mine item category information from the item review keyword documents and constructs multi-view subgraphs based on category information. Information aggregation based on node popularity is performed subsequently on each subgraph to obtain the node embeddings within each subgraph. Ultimately, based on the interaction distribution of users in each subgraph, the embeddings within multi-view subgraphs are aggregated into the final embeddings of nodes. MDCRec integrates item category information and user interests across categories into information aggregation, allowing recommendation models to capture more fine-grained relationships between items and user preferences. It can also work in tandem with other performance-enhancing techniques like contrastive learning to further boost model effectiveness. Experimental results on public real-world datasets indicate that most graph neural network recommendation models—including variants that use contrastive learning—integrated with the MDCRec information aggregation framework outperform the original popularity-based version. These models achieve varying degrees of performance gains, with average improvements of 1.75% in Recall@20 and 1.87% in NDCG@20. Our code is publicly available at https://github.com/dacilab/MDCRec.

1. Introduction

In order to mitigate the progressively severe issue of information overload, recommendation systems have been extensively deployed to offer users a personalized information filtering service. Recommendation systems center on predicting users' subsequent preferences based on their history of interactions, including purchases and clicks. As the most foundational paradigm in recommendation systems, Collaborative filtering (CF) is widely adopted for its domain-agnostic nature and reliance solely on user-item interaction data, thereby avoiding costly feature engineering on items. Based on the core assumption that users with similar interactions share similar preferences, CF methods learn latent embeddings for users and items to make predictions. Matrix factorization is one of the most dominant models [1,2], which directly models users/items as vectors and uses inner products to predict the probability of user-item interactions [3]. When interaction data for a user or item is sparse, these methods [4,5] often fail to construct effective embedded representations. Graph neural networks can incorporate neighbor nodes to aid in learning for data-sparse nodes, mitigating the data sparsity problem. Therefore, numerous existing works have integrated graph neural networks [6] into recommendation systems to boost their modeling capabilities for sparse data [7–13]. From a graph perspective, the user-item interaction in recommendation system is a bipartite graph. Both users and items are nodes in the graph, and users are linked to the items they have interacted with. Given this graph, graph neural networks can be used to capture the information hidden in interactions to enhance user/item representation learning. Numerous graph convolutional networks-based methods have demonstrated state-of-the-art recommendation results on

* Corresponding authors.
E-mail addresses: jiaxuansong1209@whu.edu.cn (J. Song), yueli0301@whu.edu.cn (Y. Li), dtclee1222@whu.edu.cn (D. Li), wxguang@whu.edu.cn (X. Wang), ruizhang8633@whu.edu.cn (R. Zhang), zhanghui0823@whu.edu.cn (H. Zhang), guangnianchenai@gmail.com (J. Chen).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.knosys.2026.115449
Received 6 July 2025; Received in revised form 4 December 2025; Accepted 29 January 2026
Available online 7 February 2026

multiple public datasets [14–18]. While recommendation models based on graph convolutional networks (GCN) can aggregate neighborhood information from user-item graph, this topology-based aggregation strategy struggles to effectively model the influence of item category features on user behavior. This is because existing methods commonly adopt an information aggregation strategy based on item popularity, and nodes with low popularity are often assigned higher weights. This is because, in the aforementioned methods, neighbor nodes are aggregated into the node through a weighted sum approach, where the weight is computed as the reciprocal of the node's historical interaction frequency. As illustrated in Fig. 1(a). The more historical interactions a node has, the lower the weight between it and other nodes. While this approach can diminish the influence of high-degree nodes during information aggregation, enabling less popular nodes to contribute information equitably, it overlooks the category correlations among items. For item nodes, some neighbor nodes that are irrelevant or have low relevance to the item node are incorporated into the learning of node representations; for user nodes, this popularity-only strategy struggles to accurately model users' genuine preferences for different categories.

Consider the information aggregation process for electronic products category item "camera" as an example of item node. Fig. 1(a) demonstrates the information aggregation process for the camera. If only popularity is considered, guitar and drum—two musical instrument items with low relevance to cameras—are aggregated into the camera's representation as second-hop neighbor nodes alongside other electronics in the same way. This is highly detrimental to camera recommendations because users are more likely to purchase a camera due to buying similar electronic products, rather than because they bought the same musical instruments. Category features of nodes are difficult to be learned accurately when the popularity-only strategy aggregating information is employed. This issue is also present within single electronic products category. This category encompasses various products, including robotic arms, servers, mobile phones, and laptops. However, compared to robotic arms and servers, mobile phones and laptops have a higher similarity to cameras. The former two are more suited for production, whereas the latter two are better suited for users' personal needs. Therefore, despite all belonging to this category, for camera information aggregation, the weights of mobile phone and laptop nodes should be higher than robotic arm and server. Calculating weights solely based on popularity would have an adverse effect on camera information aggregation.

For the user node, as depicted in Fig. 1(b), user 1 has purchased multiple electronic products and a drum. Based on the purchase history, the user exhibits a stronger preference for electronic products and also shows some interest in musical instruments. However, due to the impact of popularity on node aggregation, if the drum has low popularity, its high-hop neighbors and the drum itself will be assigned higher weights, causing items similar to the drum to be over-recommended. Conversely, if the drum has high popularity, items similar to it will be recommended less frequently, or even rarely. Both of these situations conflict with the user's actual preferences. This issue is also present within single electronic products category. User 1 may have purchased mobile phones and laptops out of personal interest, but bought a robotic arm due to company procurement needs. Compared with mobile phones and laptops, robotic arms typically have lower popularity, resulting in over-recommendation of similar electronic products, even if the user 1 has little interest in them.

In practical scenarios, manual category labeling is challenging due to the item scale. We propose MDCRec, a graph convolutional framework integrating deep clustering. First, a deep clustering module processes TF-IDF keyword documents from item reviews to generate hard and soft labels. Hard labels assign items to deterministic clusters for subgraph construction, while soft labels represent category distributions. Users' preference distributions are then derived from their interaction counts across these subgraphs.

Through this strategy, we integrate item categories and user preferences into the aggregation process. For each cluster, we compute representations via a popularity-based strategy on the subgraph, followed by a weighted sum using the category and preference distributions.

The fundamental distinction between MDCRec and existing paradigms lies in the transition from pure popularity-based aggregation to a category-aware graph aggregation framework. While pure popularity-based methods rely predominantly on structural frequency and connectivity—often overlooking underlying semantics—our category-aware strategy modulates node representations with latent semantic distributions. This mechanism ensures that the aggregation is refined by fine-grained interest clusters rather than just global interaction patterns. Consequently, MDCRec captures nuanced preferences that transcend simple connectivity, while remaining compatible with techniques like contrastive learning to further bolster effectiveness.

Experimental results across multiple public datasets and baselines validate the effectiveness of our proposed framework. The principal contributions of this article are as follows:

- We propose MDCRec, a plug-and-play graph convolutional framework that incorporates category features into aggregation to enhance representation learning.
- We develop a deep clustering module to extract category semantics and design a category-popularity aware multi-view aggregation strategy for balanced node weights.
- Experiments on Yelp and Amazon demonstrate that MDCRec consistently improves various graph convolutional models for ranking prediction.

2. Related work

2.1. Deep clustering

Clustering is a fundamental task in machine learning. It assigns instances to different groups,ensuring similar samples are in the same cluster and dissimilar ones in different clusters. However, as data grows increasingly complex, traditional clustering methods struggle to manage high-dimensional data types. With the advancement of deep learning, the concept of deep clustering has emerged, aiming to jointly optimize deep representation learning and clustering to enable efficient clustering and representation learning. Based on different ways of integrating representation learning and clustering, deep clustering can be categorized into multistage deep clustering, iterative deep clustering, and simultaneous deep clustering [19].multistage deep clustering [20–23] refers to methods where representation learning and clustering modules are optimized respectively and connected sequentially. It first learns representations (embeddings),then feeds these into classical clustering models to obtain final clustering results. Some early representative methods [20,21] utilize autoencoders for feature learning and subsequently apply k-means clustering to the learned representations. While straightforward and easy to understand, Multistage deep clustering disconnects the information flow between representation learning and clustering, allowing the final performance be affected by the limitations of both sides.Iterative deep clustering [24–28] alternately performs representation learning and clustering: it first computes clustering results based on current representations and then updates representations based on current clustering results, repeating this process. Iterative deep clustering benefits from the interplay between representation learning and clustering. However, the inaccurate results of clustering in the early training stages can lead to confusing representations, which may result in even less accurate clustering outcomes. Thus, iterative clustering models heavily need high-quality pre-training modules for representation. Simultaneous deep clustering [29–33] represents the most active research direction in deep clustering. The representation learning module and the clustering module of such models are optimized simultaneously. A representative method is DEC [34], which combines autoencoders with

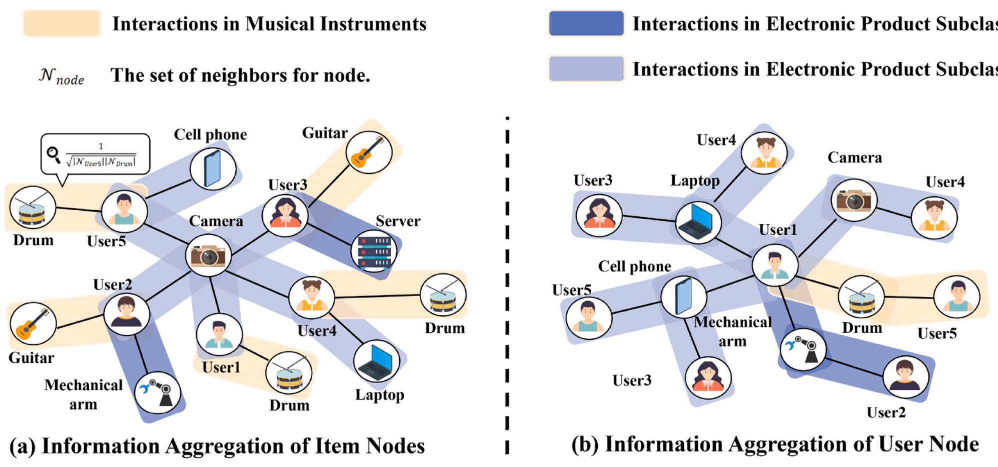
Interactions in Musical Instruments
$\mathcal{N}_{node}$ The set of neighbors for node.
Interactions in Electronic Product Subclass 2
Interactions in Electronic Product Subclass 1
$\frac{1}{\sqrt{|\mathcal{N}_{Users}||\mathcal{N}_{Drum}|}}$
Cell phone
Guitar
User3
Camera
Server
Drum
User5
User2
User4
Drum
Guitar
User1
Mechanical arm
Drum
Laptop
(a) Information Aggregation of Item Nodes
User4
User3
Laptop
Camera
User4
User1
Cell phone
User5
Mechanical arm
Drum
User5
User3
User2
(b) Information Aggregation of User Node

Fig. 1. Information aggregation for existing GCN recommendation models.

self-training strategies to optimize both clustering and representation learning. Simultaneous deep clustering has garnered the most attention due to its unified optimization, as it can learn clustering-oriented representations. However, simultaneous learning of representation and clustering may lead to an imbalance in optimization focus between the two modules, necessitating the manual setting of balancing parameters to alleviate this issue.

While existing deep clustering methods excel on general multimodal data, their direct application to recommendation scenarios is fundamentally limited. The core issue lies in their objective: they are designed to capture the dominant semantic themes within text, which in the context of user reviews, are often conflated with sentiment or specific product attributes. This results in learned representations that reflect how users feel about a product rather than what the product intrinsically is. Thus, utilizing clustering models to extract item category features from auxiliary information is a crucial problem. Inspired by existing simultaneous deep text clustering model [30], we employ the TF-IDF algorithm in the clustering module to extract keyword documents from item reviews, thereby eliminating the interference of text sentiment and other factors on item category features. We achieve effective clustering of keyword documents through contrastive representation learning combined with self-training strategy.

2.2. *Graph neural network-based recommender systems*

From a graph perspective, the user-item interaction in recommendation system is a bipartite graph. Both users and items are nodes in the graph, and users are linked to the items they have interacted with. Graph neural network (GNN) can be employed to capture user-item relationship interactions and learn effective user/item embeddings. Numerous Graph Neural Network-based recommendation system models have been introduced [8–11,35]. Among these, graph convolutional network is the most prevalent method and has been extensively applied in recommendation systems. GC-MC [36] employs a bipartite user-item graph to represent user-item interaction data, modeling user (item) nodes based on their interacted items (users). However, it disregards the original user (item) representations and only considers single-hop neighbors, thus failing to fully leverage messages propagated through the graph structure. STAR-GCN [37] utilizes the stacking of multiple independent GCN blocks to mitigate the over-smoothing issue caused by directly stacking multiple GCN layers, thereby achieving better performance than GC-MC. Both GC-MC and STAR-GCN treat neighbors' influences equally; the messages propagated depend solely on the neighbor nodes. NGCF [38] determines message propagation based on the affinity between central nodes in the graph structure. It capitalizes on the advantages of residual networks by using representations from different layers to obtain the final node embeddings, achieving remarkable improvements. PinSage [39] combines random walk strategies with graph convolution to learn node embeddings, enabling its application on large-scale data. LightGCN [12] simplifies NGCF by eliminating feature transformation and non-linear activation operations, significantly streamlining the model structure and outperforming previous models on multiple datasets. LR-GCCF [13] integrates non-linear feature propagation and residual structures into the model, thereby enhancing its performance and time efficiency. These methods leverage graph networks to learn about item neighbor nodes, thereby enriching user and item representations. To alleviate the degradation of feature representations caused by data sparsity, some works have incorporated contrastive learning into graph neural network recommendation systems [40–45].

However, these methods typically employ information aggregation strategies based solely on item popularity, lacking effective modeling of category information. This leads to two critical flaws: 1) for items, their representations become homogenized towards the average signal of their popular neighbors, obscuring their core category identity; 2) for users, the model fails to distinguish between a broad interest in popular items and a genuine, fine-grained preference for a specific category. Hence, our work employs deep clustering module to extract item category information from reviews and proposes a strategy that considers both category features and popularity. This strategy enables the model to learn item category and user interest information in a more fine-grained manner.

3. Research objectives

To clarify the research objectives of this article, we describe the limitations of existing research, the problem that need to be addressed, and the research objectives of our proposed method.

Limitations of existing research. The existing graph neural network recommendation models often use popularity-based information aggregation strategy. But this strategy leads to the introduction of some neighbor nodes that are irrelevant or have a low correlation with the user or item nodes into their information aggregation. Thereby unnecessary noise is introduced, especially when those irrelevant or less relevant neighbor nodes have a low popularity.

Problem to be addressed. The problem we try to solve is how to reduce the influence of noise from low-correlation neighbor nodes on node embedding learning and accurately identify users' true preferences for different categories.

Research objectives of our research. The research objectives of this article are as follows:

1. To develop an effective method for mining category information applicable to recommendation system scenarios from auxiliary information such as item reviews.
2. To design a new information aggregation strategy to introduce category information into the information aggregation of graph neural network recommendation models, enabling them to better learn the real preferences of users.

4. Method

4.1. Problem formulation

In this section, we formally define the multi view graph network recommendation problem based on deep clustering. Define the user set $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and the item set $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$, where M is the number of users and N is the number of items. The user-item interaction matrix is denoted as $\mathbf{R} \in \mathbb{R}^{M\times N}$. Based on implicit user feedback, $y_{uv} = 1$ indicates that user u has interacted with item v (such as clicking or purchasing), otherwise $y_{uv} = 0$. In addition to the user-item interaction matrix, we cluster the reviews of items to incorporate category information and further learn user preferences. Define the category set $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$, where K represents the number of categories. For each item, keywords are extracted from available reviews as the feature document set $\mathcal{D} = \{D_1, D_2, \ldots, D_N\}$ to represent the item category. The category distribution of item v is represented as $\mathbf{p}_v = \{p_v^1, p_v^2, \ldots, p_v^K\}$, where p_v^k denotes the probability that document D_v belongs to category C_k. If p_v^k is the maximum value in $\{p_v^1, p_v^2, \ldots, p_v^K\}$, item v will be classified into category C_k. For users, their interest preference distribution is constructed based on the items they have interacted with in different categories. The interest preference distribution of user u is defined as $\mathbf{p}_u = \{p_u^1, p_u^2, \ldots, p_u^K\}$, where p_u^k represents the number of items user u has interacted with in category C_k. Given all the above information, the recommendation of multi-view graph network based on deep clustering can be formally defined as learning a function $\hat{y}_{uv} = f(u, v, \mathbf{p}_u, \mathbf{p}_v)$, with the objective of predicting the degree to which user u is interested in item v.

4.2. The framework of MDCRec

This section presents the proposed MDCRec, a recommendation framework of multi-view graph networks based on deep clustering. Fig. 2 depicts the workflow of the MDCRec framework. The framework consists of three main modules: deep clustering on item reviews, construction of multi-cluster subgraphs, and information aggregation on multi-view subgraphs. First, for each item, we extract keywords from user reviews using TF-IDF to form feature documents, which are then clustered to obtain hard labels and soft labels. Then, based on hard labels, user-item interaction subgraphs for different categories will be constructed. For each item, its soft label serves as the clustering distribution vector. For each user, the proportion of items interacted with in different categories forms the interest distribution vector. Next, based on different subgraphs, we perform message propagation and aggregation within different clusters. Finally, according to the user's interest distribution vector and the item's clustering distribution vector, we conduct inter-cluster aggregation of user and item feature vectors from different clusters.These components will be explained in detail in the following sections.

4.3. Deep clustering on item reviews

Given that reviews contain factors like text sentiment, for each item, we first employ the TF-IDF to extract keywords from all reviews associated with the item. We construct keyword documents for each item and utilize these documents to establish item category distribution vector. We utilize the BERT model for document embedding and conduct item document clustering with contrastive learning. We first employ the pretrained BERT model (MiniLMv2 [46]) to embed all item documents and subsequently apply K-means to compute the original cluster centers $\mu_k, k \in \{1, \ldots, K\}$, which are incorporated as part of the model parameters.

To enhance the BERT model's ability to represent keyword documents using contrastive learning, we perform data augmentation on item documents. For a batch of n items, we generate two augmented documents: one by copying the item document and another by randomly removing a few words, resulting in $2n$ item documents. For an item v, we define v^1 as its document in the first augmented documents and v^2 as its document in the second augmented documents. Then, v^1 and v^2 form a positive pair for contrastive learning, while v^1 and documents of $2n-2$ other items serve as negative examples.Let z_{v^1}, z_{v^2} denote the embeddings of v^1, v^2, respectively, and z_j be the embedding of other item's document. We minimize the following contrastive learning loss function to bring z_{v^1} and z_{v^2} closer and push z_{v^1} and $z_{j\neq v^2}$ further apart:

$$L_{v^1}^{\mathrm{CL}} = -\log \frac{\exp(\mathrm{sim}(z_{v^1}, z_{v^2})/\tau)}{\sum_{j=1}^{2n} \delta_{j\neq v^1} \cdot \exp(\mathrm{sim}(z_{v^1}, z_{j^2})/\tau)}, \tag{1}$$

where, $\delta_{j\neq v^1}$ is an indicator, and τ represents the temperature hyperparameter. $\mathrm{sim}(z_{v^1}, z_{v^2})$ denotes the similarity between z_{v^1} and z_{v^2}, defined as:

$$\mathrm{sim}(z_{v^1}, z_{v^2}) = \frac{z_{v^1}^\top z_{v^2}}{\|z_{v^1}\|_2 \|z_{v^2}\|_2}. \tag{2}$$

The overall contrastive learning loss is expressed as:

$$L_{\text{Cluster-CL}} = \sum_{i=1}^{2n} \frac{L_{v^i}^{\mathrm{CL}}}{2n}. \tag{3}$$

The clustering module is learned simultaneously. Let e_v denote the embedding of item v's original document. The probability q_{vk} that item v belongs to the kth cluster is computed using the Student's t-distribution:

$$q_{vk} = \frac{(1 + \|e_v - \mu_k\|_2^2/\alpha)^{-(\alpha+1)/2}}{\sum_{k'=1}^{K}(1 + \|e_v - \mu_{k'}\|_2^2/\alpha)^{-(\alpha+1)/2}}, \tag{4}$$

where, α represents the degrees of freedom of the Student's t-distribution. We then compute the auxiliary distribution q'_{vk} and minimize the KL divergence between q_{vk} and q'_{vk} for unsupervised clustering. q'_{vk} is expressed as:

$$q'_{vk} = \frac{q_{vk}^2/f_k}{\sum_{k'} q_{vk}^2/f_{k'}}, \tag{5}$$

where, $f_k = \sum_{i=1}^{n} q_{ik},\ k = 1, \ldots, K$ represents the sum of soft clustering frequencies of all items in the kth cluster within a batch. This auxiliary distribution encourages learning high-confidence scores and overcomes biases from imbalanced clusters. We optimize the clustering distribution of item v by minimizing the following clustering loss:

$$L_v^C = \mathrm{KL}(q_v \| q'_v) = \sum_{k=1}^{K} q'_{vk} \log \frac{q'_{vk}}{q_{vk}}. \tag{6}$$

where, L_v^C denotes the clustering loss for item v, computed as the Kullback-Leibler (KL) divergence between the primary distribution q_v and the auxiliary distribution q'_v. In the summation, K is the total number of clusters, k is the index of a specific cluster. The overall clustering loss is expressed as:

$$L_{\text{Cluster-C}} = \sum_{v=1}^{N} \frac{L_v^C}{N}. \tag{7}$$

The overall loss function for the clustering module is summarized as:

$$L_{\text{Cluster}} = L_{\text{Cluster-CL}} + \eta L_{\text{Cluster-C}}, \tag{8}$$

where, η is a hyperparameter that balances contrastive learning loss and clustering loss. Through the clustering module, we obtain all clustering hard and soft labels. In the subsequent sections, we construct subgraphs under multiple cluster perspectives using hard labels and perform feature aggregation across different clusters using the soft labels.

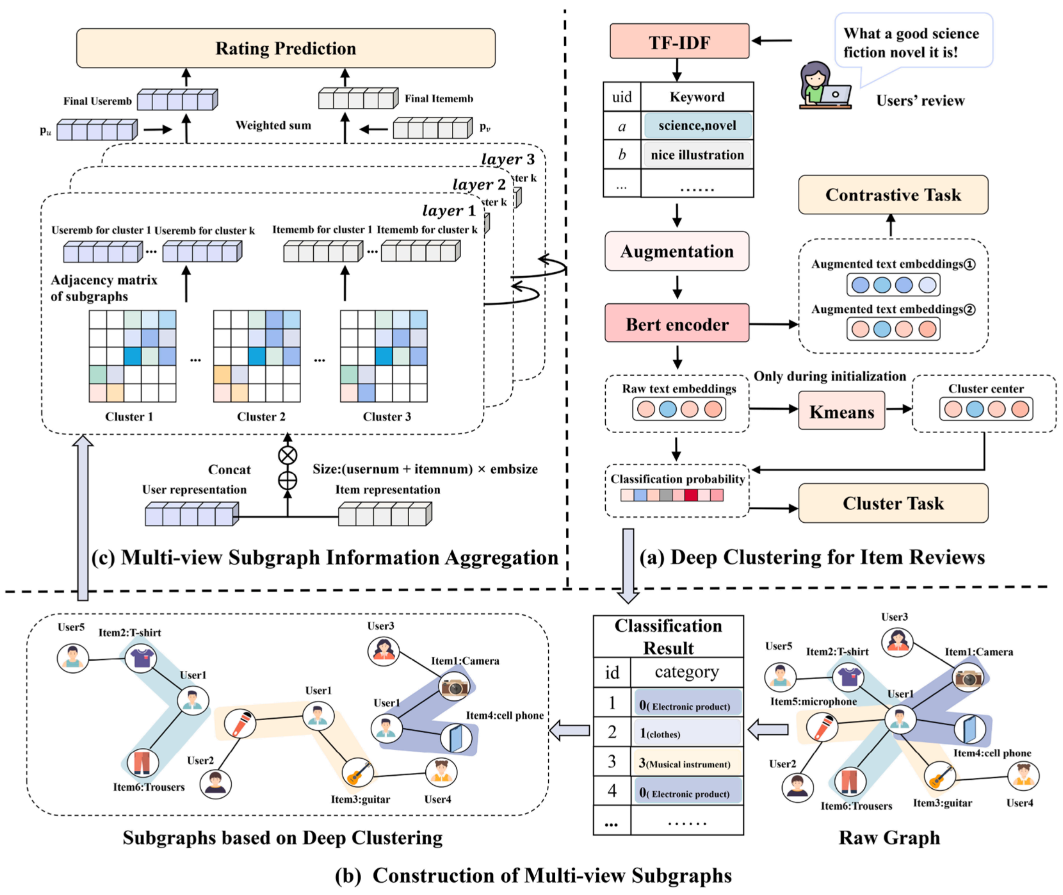

Fig. 2. An overview of MDCRec. (a): Deep clustering module is utilized to extract hard and soft labels of items from reviews; (b): Multi-view subgraphs were constructed based on hard labels; (c): After graph convolution is performed on different subgraphs, $\mathbf{p}_u$ (proportion of user's interactions in subgraphs) and $\mathbf{p}_v$ (soft labels) are used as weights to aggregate information across different clusters.

4.4. Construction of multi-cluster subgraphs

Based on the hard labels, we construct user-item interaction subgraphs for different clusters. For category $C_k \in C$, let G_k denote the corresponding graph, and $S_k = \{(u, v)|u \in U, c_v = k\}$ represent the set of all user-item interactions in this category. The user-item interaction matrix for this category is $\mathbf{R}^k \in \mathbb{R}^{M\times N}$. If there is an interaction between a user and an item in the category, $\mathbf{R}^k_{uv} = 1$; otherwise, it is 0. The adjacency matrix $\mathbf{A}^k$ for S_k is formulated as:

$$\mathbf{A}^k = \begin{pmatrix} 0 & \mathbf{R}^k \\ \mathbf{R}^{k\top} & 0 \end{pmatrix}. \tag{9}$$

When constructing the subgraph for the k^{th} cluster, we do not directly use $\mathbf{A}^k$, but combine it with adjacency matrices of other clusters to form $\mathbf{A}^k_{all}$. $\mathbf{A}^k_{all}$ is formulated as follows:

$$\mathbf{A}^k_{all} = \sigma\mathbf{A}^k + (1-\sigma)\sum_{k'=1}^{K} \delta_{k'\neq k}\cdot \mathbf{A}^{k'}, \tag{10}$$

where, σ is a hyperparameter that balances the importance of interaction edges within this category in the graph. We compute $\mathbf{A}^k_{all}$ instead of directly using $\mathbf{A}^k$ as the adjacency matrix for G_k because using $\mathbf{A}^k$ directly would result in the loss of higher-hop interaction information.

As shown in Fig. 3, there is indirect information transfer between user 1 and item 7, and between user 1 and item 8. However, the information transfer path for item 7 exists because user 1 and user 3 jointly interacted with item 1, which is not in this cluster. Therefore, for this cluster, item 7 has less influence on the user's interest features compared to item 8, but it is not without influence. Directly using $\mathbf{A}^k$ as the adjacency matrix for G_k would lead to a significant loss of such higher-hop interaction information, thereby affecting model performance. Thus, reducing the weights of edges corresponding to items not in this cluster is a better choice for modeling the interaction graph of this cluster.

4.5. Information aggregation on multi-view subgraphs

In this section, we use LightGCN [12] as the backbone for MDCRec to show the aggregation process on multi-cluster subgraphs. LightGCN employs a simple aggregation mechanism,which makes it highly efficient and significantly reduce computational costs. It mainly performs L layers of recursive aggregation for nodes.

In the l^{th} layer of each aggregation, the aggregation for multi-cluster subgraphs is described as follows:

$$\mathbf{e}^{(l+1)}_{ku} = \sum_{v\in\mathcal{N}_{ku}} \frac{1}{\sqrt{|\mathcal{N}_{ku}||\mathcal{N}_{kv}|}}\mathbf{e}^{(l)}_{v}, \tag{11}$$

$$\mathbf{e}^{(l+1)}_{kv} = \sum_{u\in\mathcal{N}_{kv}} \frac{1}{\sqrt{|\mathcal{N}_{ku}||\mathcal{N}_{kv}|}}\mathbf{e}^{(l)}_{u}, \tag{12}$$

where, $\mathbf{e}^{(l)}_{ku}$ and $\mathbf{e}^{(l)}_{kv}$ represent the embeddings of user u and item v at the lth layer in the kth cluster. $\mathcal{N}_u$ and $\mathcal{N}_v$ denote the neighbors of user u and item v, while $\mathcal{N}_{ku}$ and $\mathcal{N}_{kv}$ denote the neighbors of user u and item v within the kth cluster.

Subsequently, we sum the representations from different layers to obtain the embeddings $\mathbf{e}_{ku}$ and $\mathbf{e}_{kv}$ in the kth cluster, as follows:

$$\mathbf{e}_{ku} = \sum_{l=0}^{L}\alpha_l \mathbf{e}^{(l)}_{ku}, \quad \mathbf{e}_{kv} = \sum_{l=0}^{L}\alpha_l \mathbf{e}^{(l)}_{kv}, \tag{13}$$

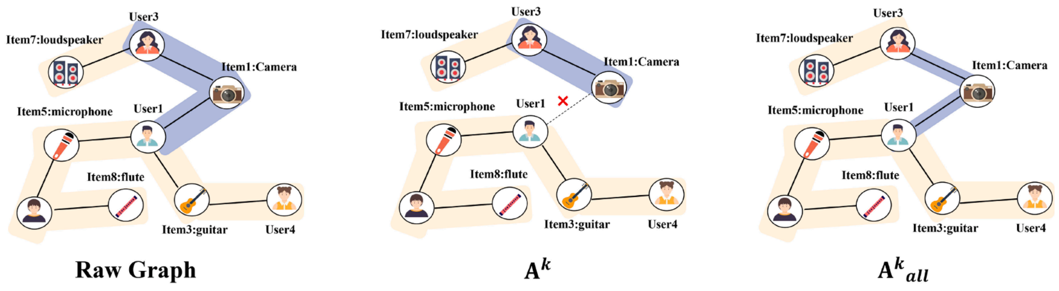

Fig. 3. Comparison of two types of subgraphs construction.

where, α_l represents the weights of representations from different layers, empirically set to $1/(L+1)$.

Then, we aggregate the user and item embeddings from different clusters to obtain the final embeddings $\mathbf{e}_u$ and $\mathbf{e}_v$:

$$\mathbf{e}_u = \sum_{k=1}^{K} \mathbf{e}_{ku} p_{ku}, \quad \mathbf{e}_v = \sum_{k=1}^{K} \mathbf{e}_{kv} p_{kv}, \tag{14}$$

where, p_{ku} represents user u's interest in the k^{th} cluster, and p_{kv} represents the probability of the item belonging to the k^{th} cluster, both of which have been regularized.p_{ku} and p_{kv} are computed using the following formulas:

$$\mathbf{p}_{ku} = \frac{p_u^k}{\sum_{k=1}^{K} p_u^k}, \quad \mathbf{p}_{kv} = \frac{p_v^k}{\sum_{k=1}^{K} p_v^k}, \tag{15}$$

where, p_v^k denotes the probability of item v belonging to category C_k, and p_u^k denotes the number of items user u has interacted with in category C_k.

We implemented the model in matrix form. Let the 0th layer embedding matrix be $\mathbf{E}^{(0)} \in \mathbb{R}^{(M+N)\times T}$, where T is the embedding size. The matrix equivalent form of LightGCN + MDCRec at the maximum layer L is:

$$\mathbf{E}_k^{(l+1)\text{temp}} = \left(\mathbf{D}^{-\frac{1}{2}} \mathbf{A}_k^{all} \mathbf{D}^{-\frac{1}{2}}\right)\mathbf{E}^{(l)}, \tag{16}$$

$$\mathbf{E}_k^{(L)} = \text{cat}\left(\mathbf{E}_{k;1:M}^{(L)\text{temp}} \circ \mathbf{P}_{ku}, \mathbf{E}_{k;M+1:M+N}^{(L)\text{temp}} \circ \mathbf{P}_{kv}\right), \tag{17}$$

$$\mathbf{E}^{(l+1)} = \sum_{k=1}^{K} \mathbf{E}_k^{(l+1)}, \tag{18}$$

where $\mathbf{D}$ is a $(M+N)\times(M+N)$ diagonal matrix where each element $\mathbf{D}_{ii}$ indicates the number of non-zero elements in the i^{th} row of the adjacency matrix $\mathbf{A}_{all}^k$. $\mathbf{E}_k^{(l+1)\text{temp}}$ is the unweighted embedding matrix for the $(l+1)^{th}$ layer of the k^{th} cluster. $\mathbf{E}_k^{(L)}$ is the weighted embedding matrix for the L^{th} layer of the k^{th} cluster. $\mathbf{P}_{ku}$ is an $M\times 1$ matrix where the u^{th} element represents user u's interest in the k^{th} cluster p_{ku}, and $\mathbf{P}_{kv}$ is an $N\times 1$ matrix where the v^{th} element represents item v's classification into the k^{th} cluster p_{kv}. cat$(\mathbf{E}_{k;1:M}^{(L)\text{temp}} \circ \mathbf{P}_{ku}, \mathbf{E}_{k;M+1:M+N}^{(L)\text{temp}} \circ \mathbf{P}_{kv})$ refers to extracting the first M rows of $\mathbf{E}_k^{(L)\text{temp}}$ for element-wise multiplication with $\mathbf{P}_{ku}$, and rows $M+1$ to $M+N$ for element-wise multiplication with $\mathbf{P}_{kv}$, then concatenating the results.

Finally, the final embedding matrix for model prediction is obtained as:

$$\mathbf{E} = \alpha_0\mathbf{E}^{(0)} + \alpha_1\mathbf{E}^{(1)} + \alpha_2\mathbf{E}^{(2)} + \ldots + \alpha_L\mathbf{E}^{(L)}. \tag{19}$$

4.6. Rating prediction

In the preceding sections, we obtained the user u's representation $\mathbf{e}_u$ and item v's representation $\mathbf{e}_v$ through message aggregation. Based on this, we can compute the likelihood score for the final item recommendation results:

$$\hat{y}(u,v) = \mathbf{e}_u^{\top}\mathbf{e}_v. \tag{20}$$

To optimize the model, we adopted the BPR loss to define the loss function:

$$L_{Rec-BPR} = \sum -\log sig\left(\left(\hat{y}(u,v^+) - \hat{y}(u,v^-)\right)\right), \tag{21}$$

where v^+ represents the items that user u interacted with during training, and v^- represents the items not observed. $sig()$ denotes the sigmoid function.

By combining the contrastive loss with the BPR loss, we optimize the following objective function to learn the model parameters:

$$\mathcal{L}_{MDCRec} = L_{Rec-BPR} + \lambda\|\theta\|_2^2, \tag{22}$$

where θ represents the model parameter set,and λ is a hyperparameter used to control the L2 regularization term. The overall optimization process for MDCRec is outlined in Algorithm 1.

Algorithm 1: The training procedure for MDCRec.

```
Input  : User set $\mathcal{U}$, Item set $\mathcal{I}$, User-item interaction data $\mathcal{R}$,
         Item review $\mathcal{D}$,Epoch number $N_e$, Batch size $N_b$.
1  Train clustering model on $\mathcal{D}$ and construct subgraphs $\mathbf{A}_{all}^k$ as
     Sections 4.3 & 4.4 ;
2  Calculate distribution of user's preference and item's category as
     $\mathbf{P_u}$ , $\mathbf{P_v}$ ;
3  for $i \leftarrow 1$ to $N_e$ do
4      Sample a mini batch from $\mathcal{R}$ ;
5      for $j \leftarrow 1$ to $N_b$ do
6          for $j \leftarrow 1$ to $N_b$ do
7              Calculate $\mathbf{e}_{ku}$ , $\mathbf{e}_{kv}$ on different subgraphs
                 // Obtain cluster-specific embeddings ;
8              $\mathbf{e}_u = \sum_{k=1}^{K} \mathbf{e}_{ku} p_{ku}$,   $\mathbf{e}_v = \sum_{k=1}^{K} \mathbf{e}_{kv} p_{kv}$ // Aggregate
                 embeddings across clusters ;
9          end
10         Calculate loss as Eq. (22) and update model parameters
             ;
11     end
12 end
Output: MDCRec
```

5. Experiments

5.1. Dataset

We conducted experiments on two public datasets: Yelp and Amazon. The details of the dataset are shown in Table 1.

- **Yelp dataset**: This dataset comprises reviews and ratings of businesses across multiple categories, including restaurants, cafes, bars, and hotels, from various global cities. Since the Yelp platform permits reviews of unexperienced items, to ensure authenticity, we selects reviews made by users with consumption records from 2016 to 2020 (interactions in Yelp Tip3) and performs 5 - core processing.

Table 1
Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Sparsity
Instruments	1429	900	10,261	0.798%
Office	4905	2420	53,258	0.449%
Music	5541	3568	64,706	0.327%
Phones	27,879	10,429	194,439	0.067%
Clothing	39,387	23,033	278,677	0.031%
Amazon-all	73,784	40,342	601,341	0.020%
Yelp	7768	18,577	217,459	0.151%

Table 2
Parameter range.

Parameter	Range
K	$[4, 6, 8, 10, 12]$
η	$[1, 5, 10]$
α	$[0.5, 1.0, 1.5]$
τ	$[0.45, 0.5, 0.55]$
ε	$[0.05, 0.1, 0.15, 0.2]$
β	$[0.05, 0.1, 0.15, 0.2]$
τ'	$[0.1, 0.15, 0.2]$
L	$[1, 2, 3]$
σ	$[0.70, 0.75, 0.80, 0.85, 0.90]$

Only users and items with a minimum of 5 interactions are retained. For each business, all associated user reviews are concatenated into a single document, and TF-IDF is applied to extract keyword features that reflect the item's category characteristics.

- **Amazon dataset**: This dataset is from Amazon 5 - core, where each user/item has at least 5 interactions/reviews. To demonstrate that MDCRec works not only for complex datasets with multi-category items but also for single-category item datasets, We chose subsets of five specific item categories: “Clothing Shoes and Jewelry”, “Cell Phones and Accessories”, “Office Products”, “Music Instruments”, “Digital Music”. These subsets are concisely termed Clothing, Phones, Office, Instruments, Music. As users exist across these subsets, we combined them into Amazon-all dataset. The training set of Amazon-all includes all the interactions in the training sets of the five subsets, and the validation and test sets are also constructed in the same way. Similarly, for each product, all user reviews are concatenated into a complete document.

Similar to most recent works, our study also uses datasets based on implicit feedback, where a label of 1 signifies positive user feedback toward item, and 0 indicates otherwise. When there is interaction between the user and an item, the implicit feedback value can be regarded as 1. On the contrary, if there is no interaction between the user and a item, then the implicit feedback value is 0. For negative samples, we randomly samples from items that user has not interacted with. All the datasets are divided into three parts—training, validation, and test sets—in an 8:1:1 ratio.

5.2. Baselines

- **BPRMF**[47]: This is a matrix factorization model based on Bayesian personalized ranking.It constructs collaborative filtering recommendations by modeling users' partial order preferences for items. It directly optimizes for personalized ranking and effectively processes implicit feedback data, which lacks negative observations.
- **NGCF**[38]: This is a graph neural network-based model. It primarily transforms user-item interaction data into graph structure and leverages GCN to learn representations of user and item. This model effectively captures high-order connectivity between users and items. It goes beyond direct interactions to uncover deeper relational insights, enhancing recommendation quality.
- **LightGCN**[12]: This is a simplified and enhanced graph convolutional network-based recommendation method, retaining only essential graph convolutional layers.
- **SGL**[40]: This is a graph contrastive learning-based recommendation method. Its core idea is to augment data on the user-item graph, generating subgraphs. It applies graph convolutional neural networks to subgraphs, extracting diverse node representations. Then, it constructs contrastive learning tasks to boost recommendation accuracy and robustness.
- **DirectAU**[48]: This is an optimization method for representation alignment and uniformity in collaborative filtering. DirectAU enhances recommendation performance by directly optimizing these two properties' learning objectives.
- **MixGCF**[49]: This is a graph neural network-based recommendation system training method. It boosts model training effectiveness through negative sample synthesis, more effectively capturing user interest preferences to enhance recommendation quality.
- **XSimGCL**[42]: This is a simple graph contrastive learning recommendation model. It adopts a simple sampling-based graph augmentation strategy during contrastive learning. This reduces computational overhead while significantly boosting recommendation performance.
- **Rankformer**[50]:This is a ranking-inspired recommendation model based on a unique graph transformer architecture. It leverages global information from all users and items, and uses specific attention weights to guide the evolution of embeddings towards improved ranking performance. It is the current state-of-the-art(SOTA) method in graph neural network recommendation.
- **DPNS**[51]:This is a negative sampling framework for graph collaborative filtering that mitigates feedback conflicts and false negatives by disentangling positive and negative feedback into different feature spaces.
- **MixRec**[52]:This is a data augmentation framework for recommender systems that mitigates data sparsity and training inefficiency by employing a dual mixing strategy for efficient sample construction.
- **ReCAFR**[53]:This is a review-centric recommendation framework that mitigates data sparsity and representation inconsistency by employing contrastive alignment to unify user, item, and review representations.

We selected three representative graph convolutional recommendation methods: NGCF, LightGCN, and XSimGCL as MDCRec's backbones. Extensive comparative experiments validated MDCRec's effectiveness.

5.3. Evaluation metrics

We have utilized two common evaluation metrics for recommendation, NDCG and Recall. NDCG evaluates the position-aware ranking performance. It is calculated as $\text{NDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p}$, where IDCG_p represents the ideal cumulative gain, and DCG_p is defined as:

$$\text{DCG}_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(1+i)}, \tag{23}$$

where rel_i denotes the relevance score of the item at rank i, and p is the cutoff length of the recommendation list.

Recall measures the proportion of relevant items successfully retrieved from the complete set of ground-truth items. It is defined as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{24}$$

where TP is the number of relevant items correctly recommended, and FN is the number of relevant items missed. Higher values for both metrics indicate better performance.

Table 3
Performance comparison for different backbones and their MDCRec variants.

Layers	Model	Clothing		Phones		Office		Instruments		Music		Amazon-all		Yelp	
		Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
1-Layer	NGCF	0.05082	0.02433	0.1237	0.06069	0.10841	0.05142	0.19138	0.08715	**0.275**	**0.14416**	0.09291	0.04632	0.07449	0.04075
	+MDCRec^{sc}	**0.05235**	**0.02511**	0.12367	0.06143	0.10622	**0.05166**	0.16264	0.07468	0.26193	0.13689	**0.09471**	**0.04676**	**0.07627**	**0.04222**
	+MDCRec	0.05164	0.02477	**0.12474**	**0.06139**	0.1079	0.04987	**0.19144**	**0.08891**	0.26127	0.1377	0.09375	0.04649	0.0758	0.04341
	LightGCN	0.05177	0.02433	0.14315	0.07106	0.12452	0.06145	0.21441	0.10266	0.29075	0.15476	0.09328	0.04578	0.08377	0.04751
	+MDCRec^{sc}	0.05631	0.02682	0.14095	0.06994	0.13073	0.06193	0.22127	0.01043	0.28426	0.15009	0.09274	0.04468	0.06837	0.03781
	+MDCRec	**0.05830**	**0.02765**	**0.14591**	**0.07347**	**0.13676**	**0.06721**	**0.22738**	**0.111**	**0.29822**	**0.15826**	**0.0985**	**0.04836**	**0.08949**	**0.05153**
	XSimGCL	0.05993	0.02865	0.15463	0.07779	0.14105	0.07065	0.22473	0.11379	0.30271	0.16712	0.10973	0.05458	0.09528	0.05573
	+MDCRec^{sc}	0.05245	0.02528	0.14462	0.07091	0.12486	0.06195	0.23065	0.10852	0.29212	0.15565	0.08708	0.0441	0.0644	0.03645
	+MDCRec	**0.06167**	**0.03003**	**0.15665**	**0.07883**	**0.14355**	**0.0723**	**0.2373**	**0.11753**	**0.3043**	**0.16766**	**0.11141**	**0.05579**	**0.09787**	**0.05676**
2-Layer	NGCF	0.05228	0.02523	0.12449	**0.06232**	0.10627	0.05096	0.18575	0.08513	**0.27646**	**0.14427**	0.09394	0.04648	0.07500	0.04200
	+MDCRec^{sc}	0.05317	0.02539	**0.12687**	0.06227	0.11038	0.05343	0.17708	0.08135	0.16393	0.08058	0.09331	0.04642	0.07285	0.04102
	+MDRec	**0.05320**	**0.02596**	0.12537	0.06196	**0.11592**	**0.0538**	**0.19676**	**0.09436**	0.26798	0.14112	**0.0942**	**0.04716**	**0.07766**	**0.04298**
	LightGCN	0.05703	0.02695	0.14695	0.07381	0.13173	0.06537	0.22936	0.11048	0.28871	0.15596	0.09663	0.04832	0.08418	0.0483
	+MDCRec^{sc}	0.05830	0.02765	0.14149	0.07001	0.12826	0.06234	0.21757	0.10268	0.28142	0.15297	0.09217	0.04445	0.08083	0.0485
	+MDRec	**0.05922**	**0.02853**	**0.14964**	**0.07489**	**0.14044**	**0.06906**	**0.2299**	**0.11165**	**0.29722**	**0.15948**	**0.10165**	**0.04983**	**0.09094**	**0.05365**
	XSimGCL	0.06729	**0.0329**	0.16258	0.08138	0.1471	**0.07642**	0.23178	0.11718	0.30725	**0.17266**	0.11714	0.0587	0.09582	0.05583
	+MDCRec^{sc}	0.05974	0.02876	0.15064	0.07383	0.13349	0.06573	0.23553	0.11481	0.29762	0.16041	0.0895	0.04222	0.08081	0.0461
	+MDCRec	**0.06759**	**0.03259**	**0.16393**	**0.08176**	**0.14851**	0.07534	**0.24197**	**0.11613**	**0.31166**	0.17119	**0.1184**	**0.05928**	**0.09705**	**0.05648**
3-Layer	NGCF	0.05256	0.0255	0.12484	0.06105	0.10609	0.04945	0.17758	0.08106	0.27739	0.14319	0.09362	0.04637	0.07756	0.04378
	+MDCRec^{sc}	0.05339	0.02581	0.12349	0.06139	0.11335	0.05505	0.16502	0.0757	0.20035	0.09063	0.09315	0.04664	0.07369	0.04211
	+MDCRec	**0.05357**	**0.02596**	**0.12533**	**0.06193**	**0.11682**	**0.05551**	**0.17974**	**0.08624**	**0.2912**	**0.15765**	**0.09449**	**0.04685**	**0.07912**	**0.04407**
	LightGCN	0.05827	0.02801	0.1485	0.0743	0.13422	0.06646	0.22955	0.11178	0.28919	0.1556	0.09881	0.04876	0.08427	0.04821
	+MDCRec^{sc}	0.05807	0.02738	0.14306	0.07056	0.13004	0.06296	0.2158	0.10017	0.28158	0.15262	0.09607	0.04678	0.07962	0.04407
	+MDCRec	**0.05974**	**0.02889**	**0.14921**	**0.07478**	**0.13790**	**0.06826**	**0.23429**	**0.11023**	**0.29335**	**0.15581**	**0.10353**	**0.05161**	**0.09266**	**0.05428**
	XSimGCL	**0.07292**	**0.03534**	0.16651	0.08306	0.15017	0.07516	0.23325	0.11599	0.30621	**0.17277**	0.11941	0.05931	0.09636	0.05578
	+MDCRec^{sc}	0.06270	0.03023	0.15217	0.07502	0.13281	0.06607	0.23493	0.11073	0.29125	0.15772	0.09795	0.04676	0.07997	0.04541
	+MDCRec	0.07068	0.03447	**0.16665**	**0.08398**	**0.15132**	**0.07591**	**0.24169**	**0.11724**	**0.31044**	0.17153	**0.12107**	**0.05995**	**0.09842**	**0.05731**

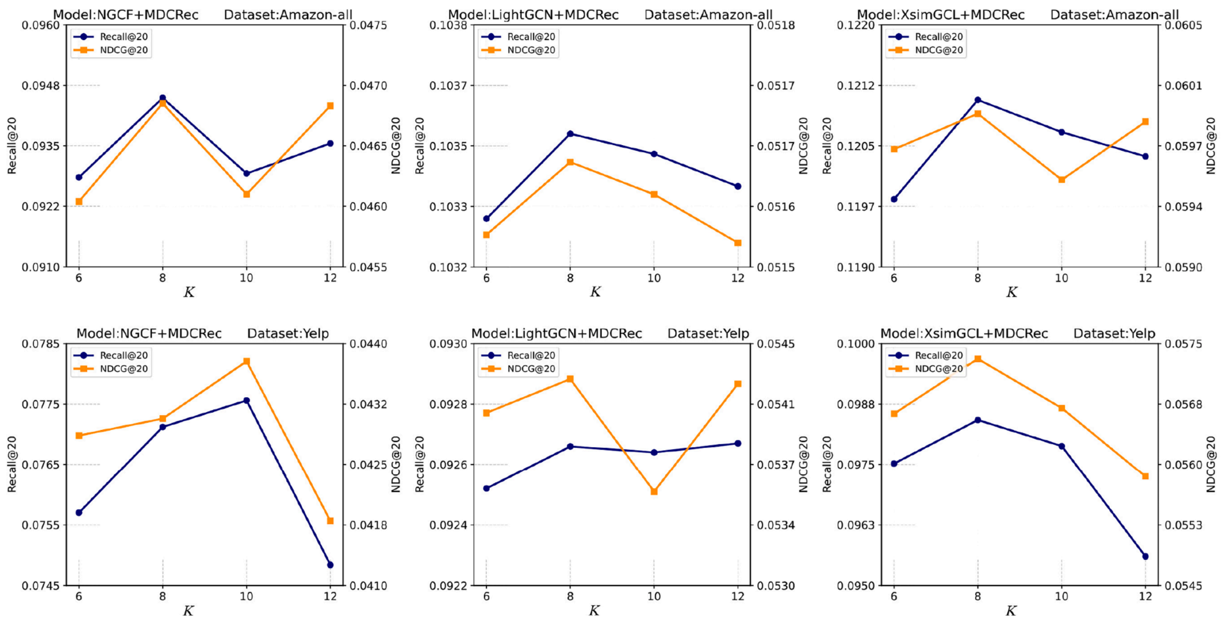

Fig. 4. Effect of clusters number K on model performance.

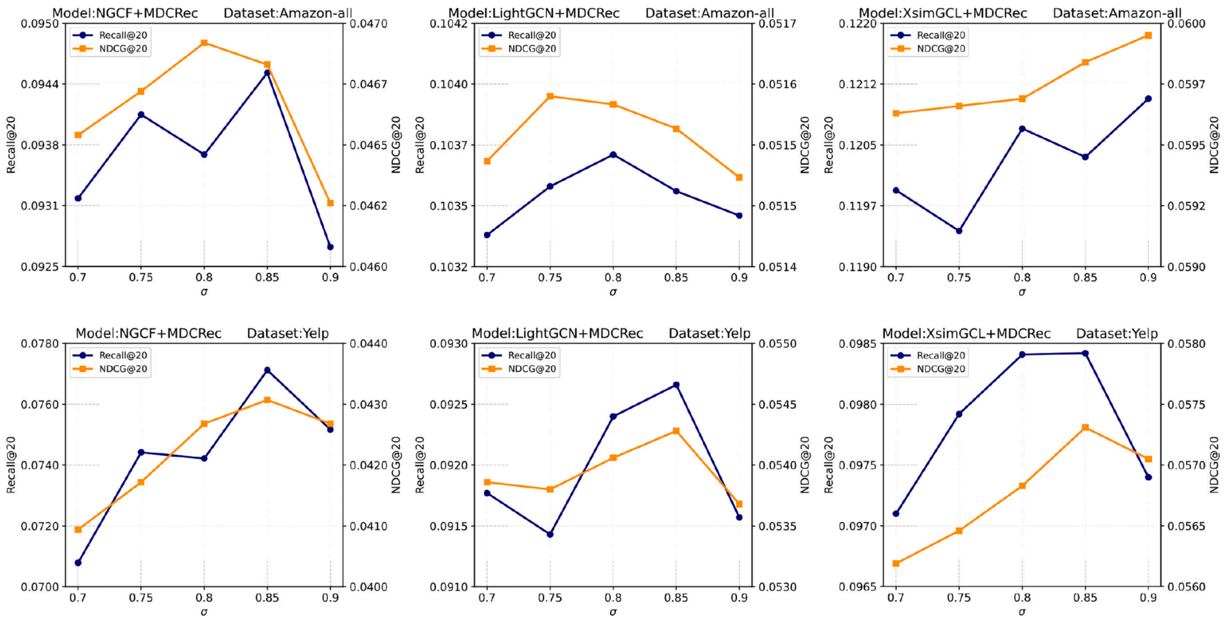

Fig. 5. Effect of subgraph construction weights σ on model performance.

5.4. Implementation details

The Experiments were performed on a server with an Intel(R) Xeon(R) Gold 6342 CPU @ 2.80GHz and an NVIDIA GeForce RTX 3090. The environment setup for all compared methods and our proposed framework includes: Ubuntu 20.04 as the operating system, Python 3.10 as the programming language, and PyTorch 2.6.0 as the deep learning framework. We implemented MDCRec and most of the baselines using the SELFREC framework [54]. For Rankformer, we ran its official code on the experimental datasets. Both MDCRec and all baselines are run on the previously mentioned datasets. For baselines, we used the optimal hyperparameters reported in their original articles and fine-tuned them via grid search. For standard settings, we initialized user and item embeddings using Xavier initialization with a dimension of 64. We used Adam for optimizing all models with a learning rate of 0.001. We also set the batch size to 2048, a common choice in many graph network recommendation studies. The keyword extraction component in MDCRec is implemented using TfidfVectorizer from the scikit-learn library, with parameters set to default configurations; keywords are selected as features from item review texts based on a TF-IDF score threshold of 0.07. The remaining parameters were determined through grid search within specified ranges, as shown in Table 2.

K: number of clusters, η: weight of contrastive learning loss in the clustering module, α: degrees of freedom for Student's t-distribution in the clustering module, τ: temperature of contrastive learning loss in the clustering module, ϵ: amplitude of noise enhancement in the recommendation module, β: weight of contrastive learning loss in the recommendation module, τ': temperature of contrastive learning loss in the recommendation module, ϵ, β, τ' are specific to models based on

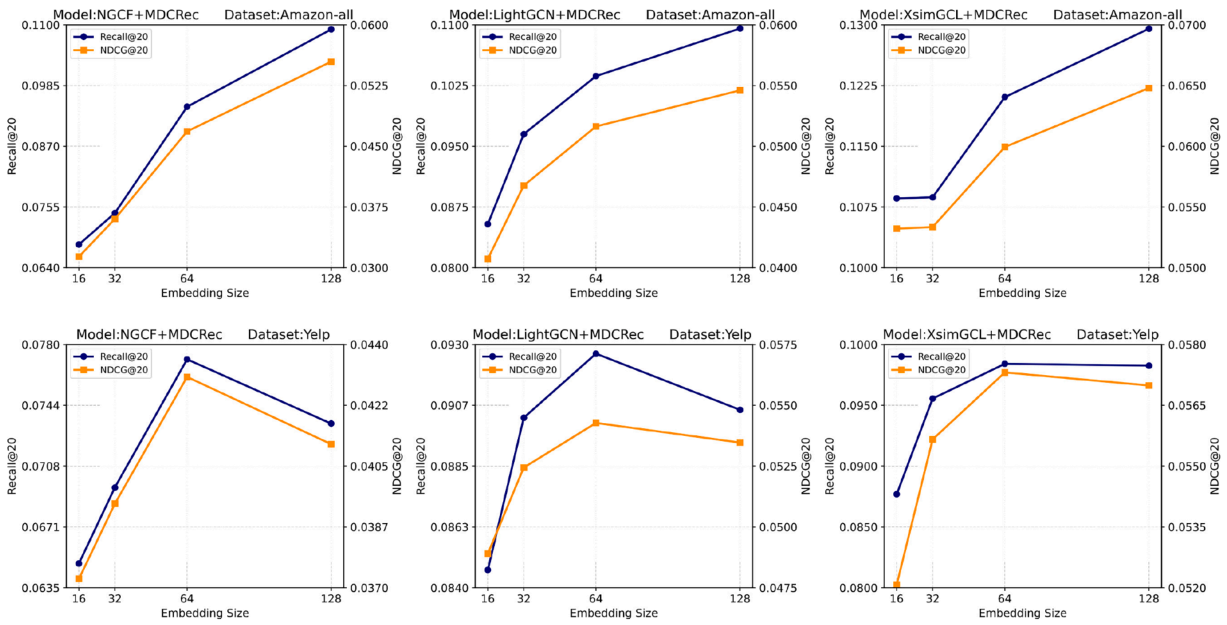

Fig. 6. Effect of embedding size on model performance.

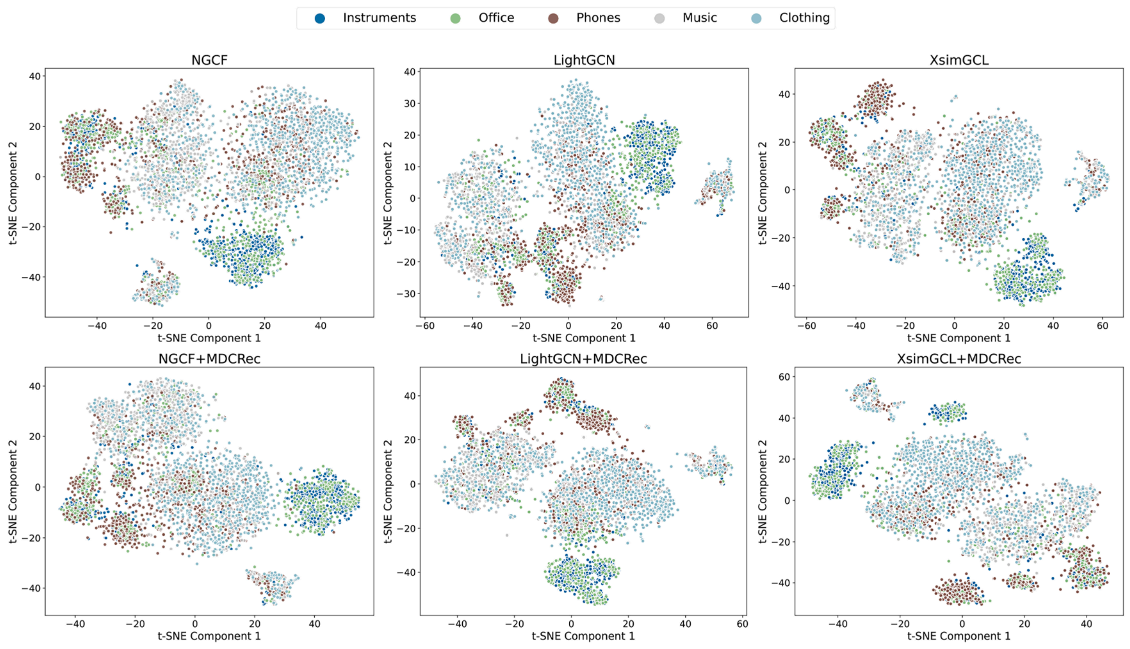

Fig. 7. Embeddings visualization on Amazon-All.

XSimGCL. L: number of graph network layers, σ: graph weight during subgraph construction.

5.5. *Performance comparison*

We compared the models with those enhanced by MDCRec's variations across different layers to evaluate MDCRec's performance. Here, +MDCRec^{sc} denotes the version using only $\mathbf{A}^k$ for constructing the k^{th} cluster's subgraph. The results are presented in Table 3. Our findings are summarized below:

1. Whether applied to subsets like Clothing or complete datasets like Amazon All and Yelp, most models with MDCRec showed improvements in Recall@20 and NDCG@20 compared to the original models, thus proving the effectiveness and universality of our proposed MDCRec framework. By integrating clustering information into the aggregation process of graph neural network, recommendation models can learn item categories and user interests information better. The MDCRec framework is applicable not only to complex datasets with diverse item categories but also effective for single-category recommendation.
2. However, in experiments with these three base models, adding $+MDCRec^{sc}$ resulted in decreased performance across multiple datasets for most models, even underperforming compared to the original base models sometimes. This suggests that directly using $\mathbf{A}^k$ as the adjacency matrix for G_k is not optimal. This phenomenon may stem from the significant loss of high-hop interaction information due to this graph construction approach. Given the inherent sparsity of data in recommendation systems—where user interac-

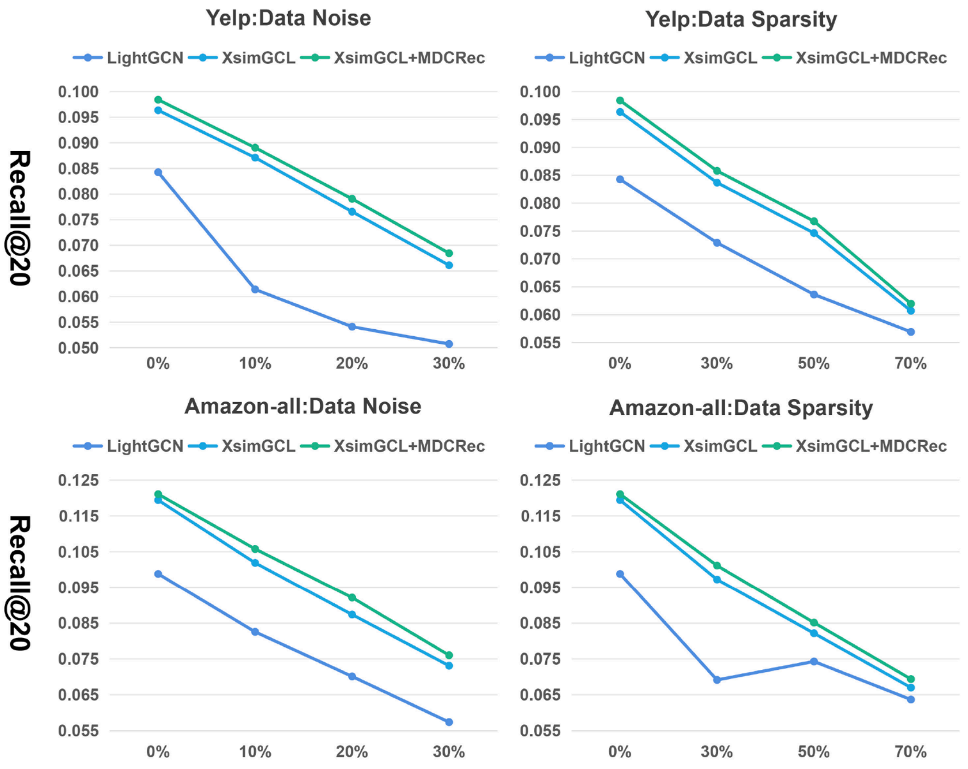

Fig. 8. Robustness evaluation with respect to data noise and sparsity.

tion histories are limited compared to the vast number of items to recommend—even the loss of some high-hop interaction information can greatly affect model performance.

3. With the integration of MDCRec, classic graph convolutional recommendation models such as NGCF and LightGCN demonstrated performance improvements, confirming that combining category information with popularity effectively enhances the model's information aggregation process. Moreover, recommendation models that incorporate graph contrastive learning, such as XSimGCL, also experienced performance improvements after integrating MDCRec. This shows that MDCRec can effectively integrate with various modules, highlighting its versatility in improving the performance of GCN-based recommendation models.

To further explore the effectiveness of MDCRec, we compared it against classical matrix factorization model and other recent graph-based recommendation models on the same datasets. Recommendation methods based on large language models are not used for comparison. This is because MDCRec is a graph neural network recommendation framework that improves the calculation of node weights for information aggregation. The focus of the comparison should be placed on the recommendation model centered on graph neural networks, so as to evaluate the effectiveness of our method more accurately. Based on the results from Table 4, MDCRec (using XsimGCL as the backbone) has demonstrated performance close to that of SOTA methods on most datasets, outperforming the latest method Rankformer on several datasets such as Amazon-all and Yelp. Even on these datasets, the performance of XsimGCL as the backbone lags behind other models. This experimental result strongly demonstrates the effectiveness of MDCRec. However, on the Clothing dataset, MDCRec paradoxically diminished XsimGCL's performance. We attribute this setback to the consistent reflection of users' clothing preferences across different subcategories in the Clothing dataset. Over-categorization of clothing items may actually impair the model's capability to comprehend user preferences. On the Office and Music datasets, although MDCRec demonstrated an excellent recall rate, it performed poorly on NDCG.We attribute the reason to the fact that the embeddings learned by MDCRec for items of the same category are more similar. Although it improves the recall rate of long-tail items under each category, it also increases the difficulty of ranking.

5.6. Parametric analysis

In this section, we investigate the influence of key parameters in MDCRec on model performance using two comprehensive datasets: Yelp and Amazon-all.

5.6.1. Effect of the number of clusters K on model performance

To examine the effect of clusters number K on performance, we varied K across [4,6,8,10,12]. The results are depicted in Fig. 4, which indicates that as K increases, model performance on both datasets tends to rise initially and then decline. This trend is attributed to the structure of data within clusters. When K is low, each cluster contains more items across multiple categories, leading to substantial feature variance within clusters. As K increases, the features within clusters become more homogeneous, aiding the model in learning more precise feature representations. However, when K is too high, each cluster contains fewer items, leading to overly fragmented subgraphs, which can degrade the model's recommendation capabilities.

5.6.2. Effect of subgraph construction weights σ on model performance

To examine the effect of subgraph weight σ on model performance, σ was varied across [0.7,0.75,0.8,0.85,0.9]. The results are shown in Fig. 5, which shows that model performance generally increases and then decreases with changes in σ, suggesting an optimal σ value. This trend is associated with the balance of information propagation during subgraph construction. When σ is low, the model incorporates excessive indirect information via paths involving different item categories; this information includes features from less relevant nodes, disrupting the model's learning and reducing performance. As σ increases, the model increasingly relies on direct information within clusters, aiding in learning more accurate feature representations. However, when σ is too high,

Table 4
Performance comparison for different GNN-based recommendation models.

Model	Clothing		Phones		Office		Instruments		Music		Amazon-all		Yelp	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
BPRMF	0.03277	0.01556	0.11351	0.05637	0.10378	0.05099	0.12962	0.06475	0.26745	0.14278	0.07689	0.03809	0.00727	0.004096
NGCF	0.05256	0.02550	0.12484	0.06105	0.10609	0.04945	0.17758	0.08106	0.27739	0.14319	0.09362	0.04637	0.07756	0.04378
NGCF + MDCRec	0.05357	0.02596	0.12533	0.06193	0.11682	0.05551	0.17974	0.08624	0.29120	0.15765	0.09449	0.04685	0.07912	0.04407
LightGCN	0.05827	0.02801	0.14850	0.07430	0.13422	0.06646	0.22955	0.11178	0.28919	0.15560	0.09881	0.04876	0.08427	0.04821
LightGCN + MDCRec	0.05974	0.02889	0.14921	0.07478	0.13790	0.06826	0.23429	0.11023	0.29335	0.15581	0.10353	0.05161	0.09266	0.05428
SGL	0.06187	0.03036	0.15891	0.08051	0.14155	0.07123	0.22988	0.11609	0.30688	0.17145	0.11847	0.05930	0.09618	0.55220
DirectAU	0.07397	0.03542	0.16606	0.08331	0.14382	0.07076	0.23926	0.11320	0.30252	0.16573	0.11946	0.05829	0.08599	0.05041
MixGCF	0.07121	0.03412	0.16504	0.08160	0.14590	0.07693	0.23924	0.11702	0.29206	0.16512	0.11918	0.05877	0.09559	0.05884
XSimGCL	0.07292	0.03534	0.16561	0.08306	0.15017	0.07516	0.23325	0.11599	0.30621	0.17277	0.11941	0.05931	0.09636	0.05578
DPNS	0.07044	0.03441	0.16659	0.08311	0.14923	**0.07699**	0.23218	0.11401	0.30218	0.17059	0.11955	0.05833	0.09668	0.05608
MixRec	0.07358	0.03535	**0.16727**	0.08385	0.14940	0.07632	0.23305	0.11410	0.30918	**0.17324**	0.11953	0.05808	0.09661	0.05704
Rankformer	0.07417	**0.03634**	0.16706	0.08392	0.15068	0.07641	0.23629	0.11436	0.30444	0.17286	0.11849	0.05868	0.09723	0.05649
ReCAFR	**0.07456**	0.036191	0.16720	**0.08405**	0.14920	0.07572	0.23415	0.11530	0.31007	0.17212	0.12003	0.05908	0.09688	0.05654
XSimGCL + MDCRec	0.07068	0.03447	0.16665	0.08398	**0.15132**	0.07591	**0.24169**	**0.11724**	**0.31044**	0.17153	**0.12107**	**0.05995**	**0.09842**	**0.05731**

the model overly focuses on direct interactions within clusters, ignoring indirect information via different item categories, preventing the model from fully capturing complex user-item relationships.

5.6.3. *Effect of embedding size on model performance*

To examine the effect of embedding size on model performance, embedding size was varied across [16, 32, 64, 128]. The results are shown in Fig. 6, which indicates that as embedding size increases, model performance consistently improves on Amazon-all. This may be because the large dataset requires more parameters to effectively model user and item features, necessitating a larger embedding size. However, due to the increased space and time cost of larger embedding sizes, we opted for an embedding size of 64 in our experiments. In contrast, on Yelp, model performance typically rises initially and then declines. This suggests that with an excessive number of model parameters, the information provided by the recommendation set is inadequate for effectively updating and optimizing numerous latent factors, resulting in underfitting.

5.7. *Embeddings visualization*

In this section, we visualize embeddings of the baselines NGCF, LightGCN, and XSimGCL, as well as the enhanced models NGCF + MDCRec, LightGCN + MDCRec, and XSimGCL + MDCRec. This analysis aims to explore how the inclusion of clustering information influences the representation learning of items. In the Amazon-All dataset, we randomly sampled 500 items from each subclass. We then projected the representations learned by various CGCN variants into a 2D space using t-SNE to generate a 2D feature distribution map. From the Fig. 7, it's evident that whether models without contrastive learning like NGCF and LightGCN,or model like XSimGCL with a contrastive learning module, After the addition of MDCRec, items of the same category are more obviously clustered in their learned representations.The distances between clusters are relatively uniform, but there's a significant reduction in intra-cluster distances, indicating more compact clusters. This suggests that the addition of the clustering information aggregation module leads the model to learn more similar representations for items within the same category, highlighting their categorical attributes. Consequently, similar items are more likely to be recommended to users with relevant preferences, facilitating the discovery of long-tail items within a category and enhancing the model's performance.

5.8. *Robustness evaluation*

5.8.1. *Data noise*

To evaluate the model's robustness against noise, we simulated data noise scenarios on the Amazon-all and Yelp datasets by replacing 10%, 20%, and 30% of the original interactions with randomly generated noisy interactions, respectively. As shown in the left part of Fig. 8, our proposed method, MDCRec, exhibits strong noise resistance. Specifically, on both the Amazon-all and Yelp datasets, as the noise level increases, the Recall@20 performance of XsimGCL + MDCRec consistently surpasses that of the original XsimGCL model and the foundational LightGCN model. Furthermore, its rate of performance degradation remains consistent with that of XsimGCL. This indicates that the performance enhancement mechanism of MDCRec possesses considerable noise resilience, enabling it to boost the performance of the base model in the presence of data noise.

5.8.2. *Data sparsity*

To assess the model's performance under data sparsity, we simulated sparse data scenarios by randomly removing 30%, 50%, and 70% of the interaction records for each user. We then compared the Recall@20 performance on the Amazon-all and Yelp datasets. As shown in the right part of Fig. 8, XsimGCL + MDCRec consistently outperformed both XsimGCL and LightGCN across all sparsity levels. Even in the extreme sparsity scenario where 70% of each user's interactions were removed, XsimGCL + MDCRec retained its capacity to enhance the original base model. This advantage stems from MDCRec's integration of a deep clustering module, which learns categorical information to augment the information aggregation process of the GCN, thereby improving the base model's performance in data-sparse scenarios.

6. Discussion

6.1. Results analysis

The MDCRec framework aims to incorporate cluster module to model the impact of item category features on user behavior in the information aggregation process, thereby enhancing the accuracy of recommendation predictions. Through comparative experiments with multiple baseline models, we have demonstrated the general effectiveness of MDCRec. Whether on single-category subsets or multi-category full sets, the graph neural network recommendation models enhanced by MDCRec can more accurately capture user intentions and preferences, outperforming existing methods in most evaluation metrics. The visualization of learned features further reveals the mechanism behind MDCRec's performance improvement by showing that items in the same category learn more similar features after incorporating the clustering-based information aggregation module. Through hyperparameter adjustment, models enhanced by MDCRec achieve remarkable results across datasets.

The experimental results highlight the following advantages of our work:

1. By integrating deep clustering module, MDCRec effectively extracts explicit item category information and user interest preferences and incorporates them into the information aggregation process. Compared to the original version that only uses popularity-based strategy, most graph neural network recommendation models with the MDCRec framework for information aggregation have demonstrated varying degrees of performance improvement.
2. The visualization of learned features indicates that the model using MDCRec can better learn the category correlation of items. Consequently, similar items are more likely to be recommended to users with relevant preferences, enhancing the possibility of long-tail projects being recommended within a category.
3. The modular design of MDCRec allows it to be applicable to most existing graph neural network recommendation models. Whether it is traditional models like NGCF and LightGCN without contrastive learning, or models like XSimGCL with contrastive learning modules, their performance has been improved after adding MDCRec. This generality enables MDCRec to be conveniently incorporated into other graph neural network models to enhance their performance.

6.2. The limitations of MDCRec

Despite its demonstrated advantages, the MDCRec framework exhibits several limitations that present opportunities for future improvement. The primary drawbacks are not merely operational but also conceptual, pointing to fundamental constraints in its current architecture.

1. Limitation in Cold-Start Scenarios. The reliance on item review text leads to poor performance in cold-start scenarios. New items lack the textual data required for the deep clustering module, which prevents their category assignment and consequently hinders the generation of meaningful embeddings.
2. Operational Inefficiency and Decoupled Training. The two-stage pipeline (clustering followed by recommendation) is inefficient. It increases computational overhead and complicates model deployment and tuning, hindering rapid iteration and real-time updates.
3. Inability to Leverage Implicit Category Signals. By relying solely on explicit text, MDCRec overlooks rich, implicit category signals inherent in the user-item interaction graph. This prevents the discovery of behavior-based clusters, limiting the granularity of learned user preferences.
4. The Challenge of End-to-End Integration. The most critical limitation is the lack of an end-to-end learning mechanism. This prevents the joint optimization of category discovery and recommendation, missing the opportunity for these two tasks to mutually reinforce each other.

6.3. Theoretical and practical implications

Theoretically, the proposed MDCRec framework leverages deep clustering to mine item category information, combines it with node popularity, and constructs multi-view subgraphs to more accurately capture user-item interactions and preferences. This addresses the limitations of traditional popularity-based methods, which overlook category correlations among items and thus impair node representation learning. Practically, MDCRec improves recommendation accuracy by modeling users' true preferences across multiple categories and item types, which traditional methods struggle to achieve. Its integration into existing systems as a node weight calculation strategy is straightforward, offering significant benefits such as enhanced user satisfaction and more precise recommendations for platforms like e-commerce.

6.4. Prospects for future work

Based on the identified limitations of MDCRec, future research can be directed towards several promising avenues to further enhance its performance and applicability.

1. Mitigating the Cold Start Problem. Future work should focus on developing strategies to handle new items without review text. This could involve leveraging side information such as item attributes, visual content, or initializing embeddings through transfer learning from other domains, ensuring that new items can be effectively categorized and recommended.
2. Leveraging Implicit Category Signals. Future work should focus on designing mechanisms to automatically discover latent item clusters from the user-item interaction graph itself. This would reduce the dependency on external textual data and capture behavior-based categories, leading to a more robust and fine-grained understanding of user preferences.
3. Achieving End-to-End Integration. The most critical direction is to develop an end-to-end learning framework that unifies clustering and recommendation. This would allow for joint optimization, where the discovered categories are directly tailored for and mutually reinforced by the recommendation objective, eliminating the inefficiency of the two-stage pipeline.

CRediT authorship contribution statement

Jiaxuan Song: Writing – review & editing, Writing – original draft, Software, Formal analysis, Data curation, Conceptualization; **Yue Li:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Data curation, Conceptualization; **Duantengchuan Li:** Writing – review & editing, Writing – original draft, Supervision, Software, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization; **Xiaoguang Wang:** Writing – review & editing, Project administration, Methodology, Funding acquisition, Conceptualization; **Rui Zhang:** Writing – review & editing, Methodology, Conceptualization; **Hui Zhang:** Writing – review & editing, Visualization, Data curation, Conceptualization; **Jinsong Chen:** Writing – review & editing, Methodology, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (Nos. 62507035, 62276213), the Major Project of the National Social Science Fund of China (No. 21&ZD334) and the Peng Feizhang Library Science Development Fund Project of Wuhan University.

References

[1] H. Liu, C. Zheng, D. Li, X. Shen, K. Lin, J. Wang, Z. Zhang, Z. Zhang, N.N. Xiong, EDMF: efficient deep matrix factorization with review feature learning for industrial recommender system, IEEE Trans. Ind. Inf. 18 (7) (2022) 4361–4371.
[2] B. Liu, D. Li, J. Wang, Z. Wang, B. Li, C. Zeng, Integrating user short-term intentions and long-term preferences in heterogeneous hypergraph networks for sequential recommendation, Inf. Process. Manage. 61 (3) (2024) 103680.
[3] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37. https://doi.org/10.1109/MC.2009.263
[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, WWW '17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2017, p. 173–182. https://doi.org/10.1145/3038912.3052569
[5] Y. Tay, L. Anh Tuan, S.C. Hui, Latent relational metric learning via memory-based attention for collaborative ranking, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, p. 729–739. https://doi.org/10.1145/3178876.3186154
[6] F. Wu, D. Li, K. Lin, H. Zhang, Efficient nodes representation learning with residual feature propagation, in: Advances in Knowledge Discovery and Data Mining, 2021, pp. 156–167.
[7] D. Li, J. Lu, Z. Wang, J. Wang, X. Wang, F. Shi, Y. Liu, Recommender system based on noise enhancement and multi-view graph contrastive learning, Appl. Soft Comput. 177 (2025) 113220. https://doi.org/10.1016/j.asoc.2025.113220
[8] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, T.-S. Chua, Disentangled graph collaborative filtering, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 1001–1010. https://doi.org/10.1145/3397271.3401137
[9] J. Yu, H. Yin, J. Li, Q. Wang, N.Q.V. Hung, X. Zhang, Self-supervised multi-channel hypergraph convolutional network for social recommendation, in: Proceedings of the Web Conference 2021, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 413–424. https://doi.org/10.1145/3442381.3449844
[10] J. Yu, H. Yin, J. Li, M. Gao, Z. Huang, L. Cui, Enhancing social recommendation with adversarial graph convolutional networks, IEEE Trans. Knowl. Data Eng. 34 (8) (2022) 3727–3739. https://doi.org/10.1109/TKDE.2020.3033673
[11] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'19/IAAI'19/EAAI'19, AAAI Press, 2019. https://doi.org/10.1609/aaai.v33i01.3301346
[12] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, 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, SIGIR '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 639–648. https://doi.org/10.1145/3397271.3401063
[13] L. Chen, L. Wu, R. Hong, K. Zhang, M. Wang, Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach, Proc. AAAI Conf. Artif. Intell. 34 (01) (2020) 27–34. https://ojs.aaai.org/index.php/AAAI/article/view/5330. https://doi.org/10.1609/aaai.v34i01.5330
[14] X. Mao, Y. Liu, L. Qi, L. Duan, X. Xu, X. Zhang, W. Dou, A. Beheshti, X. Zhou, Cluster-driven personalized federated recommendation with interest-aware graph convolution network for multimedia, in: Proceedings of the 32nd ACM International Conference on Multimedia, MM '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 5614–5622. https://doi.org/10.1145/3664647.3680788
[15] Q. Zhou, J. Wu, H. Dai, G. Yang, Y. Zhang, An intelligent ride-sharing recommendation method based on graph neural network and evolutionary computation, IEEE Trans. Intell. Transp. Syst. 26 (1) (2025) 569–578. https://doi.org/10.1109/TITS.2024.3485985
[16] E. Elahi, S. Anwar, M. Al-kfairy, J.J.P.C. Rodrigues, A. Ngueilbaye, Z. Halim, M. Waqas, Graph attention-based neural collaborative filtering for item-specific recommendation system using knowledge graph, Expert Syst. Appl. 266 (2025) 126133. https://doi.org/10.1016/j.eswa.2024.126133
[17] P. Zhang, Z. Niu, R. Ma, F. Zhang, Multi-view graph contrastive representation learning for bundle recommendation, Inf. Process. Manage. 62 (1) (2025) 103956. https://doi.org/10.1016/j.ipm.2024.103956
[18] S. Lei, X. Chang, Z. Yu, D. He, C. Huo, J. Wang, D. Jin, Feature-structure adaptive completion graph neural network for cold-start recommendation, Proc. AAAI Conf. Artif. Intell. 39 (11) (2025) 12022–12030. https://ojs.aaai.org/index.php/AAAI/article/view/33309. https://doi.org/10.1609/aaai.v39i11.33309
[19] J. Singh, D. Singh, A comprehensive review of clustering techniques in artificial intelligence for knowledge discovery: taxonomy, challenges, applications and future prospects, Adv. Eng. Inf. 62 (2024) 102799. https://doi.org/10.1016/j.aei.2024.102799
[20] P. Huang, Y. Huang, W. Wang, L. Wang, Deep embedding network for clustering, in: 2014 22nd International Conference on Pattern Recognition, 2014, pp. 1532–1537. https://doi.org/10.1109/ICPR.2014.272
[21] F. Tian, B. Gao, Q. Cui, E. Chen, T.-Y. Liu, Learning deep representations for graph clustering, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI'14, AAAI Press, 2014, p. 1293–1299.
[22] P. Ji, T. Zhang, H. Li, M. Salzmann, I. Reid, Deep subspace clustering networks, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 23–32.
[23] J. Huang, S. Gong, Deep clustering by semantic contrastive learning, in: 33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21–24, 2022, BMVA Press, 2022. https://bmvc2022.mpi-inf.mpg.de/0039.pdf.
[24] C. Song, F. Liu, Y. Huang, L. Wang, T. Tan, Auto-encoder based data clustering, in: J. Ruiz-Shulcloper, G. Sanniti di Baja (Eds.), Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 117–124.
[25] B. Yang, X. Fu, N.D. Sidiropoulos, M. Hong, Towards K-means-friendly spaces: simultaneous deep learning and clustering, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, JMLR.org, 2017, p. 3861–3870.
[26] J. Zhang, C.-G. Li, C. You, X. Qi, H. Zhang, J. Guo, Z. Lin, Self-supervised convolutional subspace clustering network, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
[27] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, L. Van Gool, SCAN: learning to classify images without labels, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), Computer Vision – ECCV 2020, Springer International Publishing, Cham, 2020, pp. 268–285.
[28] C. Niu, H. Shan, G. Wang, SPICE: semantic pseudo-labeling for image clustering, IEEE Trans. Image Process. 31 (2022) 7264–7278. https://doi.org/10.1109/TIP.2022.3221290
[29] K. Do, T. Tran, S. Venkatesh, Clustering by maximizing mutual information across views, in: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 9908–9918. https://doi.org/10.1109/ICCV48922.2021.00978
[30] D. Zhang, F. Nan, X. Wei, S.-W. Li, H. Zhu, K. McKeown, R. Nallapati, A.O. Arnold, B. Xiang, Supporting clustering with contrastive learning, in: K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, Y. Zhou (Eds.), Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021, pp. 5419–5430. https://doi.org/10.18653/v1/2021.naacl-main.427
[31] M. Shang, C. Liang, J. Luo, H. Zhang, Incomplete multi-view clustering by simultaneously learning robust representations and optimal graph structures, Inf. Sci. 640 (2023) 119038. https://www.sciencedirect.com/science/article/pii/S0020025523006230. https://doi.org/10.1016/j.ins.2023.119038
[32] W. Doo, H. Kim, Simultaneous deep clustering and feature selection via K-concrete autoencoder, IEEE Trans. on Knowl. and Data Eng. 36 (6) (2024) 2629–2642. https://doi.org/10.1109/TKDE.2023.3323580
[33] W. Cao, Z. Zhang, M. Li, Z. Yu, Advancing deep clustering through the synergy of contrastive learning and capsule-GAN, Knowl. Syst. 330 (2025) 114551. https://doi.org/10.1016/j.knosys.2025.114551
[34] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: M.F. Balcan, K.Q. Weinberger (Eds.), Proceedings of the 33rd International Conference on Machine Learning, 48 of *Proceedings of Machine Learning Research*, PMLR, New York, New York, USA, 2016, pp. 478–487. https://proceedings.mlr.press/v48/xieb16.html.
[35] W. Ma, W. Chen, L. Lu, X. Fan, Integrating learners' knowledge background to improve course recommendation fairness: a multi-graph recommendation method based on contrastive learning, Inf. Process. Manage. 61 (4) (2024) 103750. https://doi.org/10.1016/j.ipm.2024.103750
[36] C. Su, M. Chen, X. Xie, Graph convolutional matrix completion via relation reconstruction, in: Proceedings of the 2021 10th International Conference on Software and Computer Applications, ICSCA '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 51–56. https://doi.org/10.1145/3457784.3457792
[37] J. Zhang, X. Shi, S. Zhao, I. King, STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI'19, AAAI Press, 2019, p. 4264–4270.
[38] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'19, Association for Computing Machinery, New York, NY, USA, 2019, p. 165–174. https://doi.org/10.1145/3331184.3331267
[39] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of

the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 974–983. https://doi.org/10.1145/3219819.3219890

[40] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, X. Xie, Self-supervised graph learning for recommendation, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 726–735. https://doi.org/10.1145/3404835.3462862

[41] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, Q.V.H. Nguyen, 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, SIGIR '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 1294–1303. https://doi.org/10.1145/3477495.3531937

[42] J. Yu, X. Xia, T. Chen, L. Cui, N.Q.V. Hung, H. Yin, XSimGCL: towards extremely simple graph contrastive learning for recommendation, IEEE Trans. Knowl. Data Eng. 36 (2) (2024) 913–926. https://doi.org/10.1109/TKDE.2023.3288135

[43] C. Zeng, Y. Qian, Y. Tang, H^{2}GRL: a heterogeneous-homogeneous graph contrastive learning framework for long-tail recommendation, Expert Syst. Appl. 293 (2025) 128727. https://doi.org/10.1016/j.eswa.2025.128727

[44] K. Shi, Y. Zhang, M. Zhang, K. Xiao, X. Hou, Z. Li, Noise-enhanced graph contrastive learning for multimodal recommendation systems, Knowl. Syst. 324 (2025) 113766. https://doi.org/10.1016/j.knosys.2025.113766

[45] J. Liu, W. Wang, B. Yi, X. Shen, H. Zhang, Contrastive multi-interest graph attention network for knowledge-aware recommendation, Expert Syst. Appl. 255 (2024) 124748. https://doi.org/10.1016/j.eswa.2024.124748

[46] W. Wang, H. Bao, S. Huang, L. Dong, F. Wei, MiniLMv2: multi-head self-attention relation distillation for compressing pretrained transformers, in: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, Association for Computational Linguistics, Online, 2021, pp. 2140–2151. https://doi.org/10.18653/v1/2021.findings-acl.188

[47] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI '09, AUAI Press, Arlington, Virginia, USA, 2009, p. 452–461.

[48] C. Wang, Y. Yu, W. Ma, M. Zhang, C. Chen, Y. Liu, S. Ma, Towards representation alignment and uniformity in collaborative filtering, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 1816–1825. https://doi.org/10.1145/3534678.3539253

[49] T. Huang, Y. Dong, M. Ding, Z. Yang, W. Feng, X. Wang, J. Tang, MixGCF: an improved training method for graph neural network-based recommender systems, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 665–674. https://doi.org/10.1145/3447548.3467408

[50] S. Chen, S. Han, J. Chen, B. Hu, S. Zhou, G. Wang, Y. Feng, C. Chen, C. Wang, RankFormer: a graph transformer for recommendation based on ranking objective, in: Proceedings of the ACM on Web Conference 2025, WWW '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 3037–3048. https://doi.org/10.1145/3696410.3714547

[51] H. Li, X. Zhang, H. Weng, Y. Shen, K. Cai, D. Wang, Z. Qin, S. Deng, Disentangled progressive negative sampling for graph collaborative filtering recommendation, Knowl. Syst. 327 (2025) 114133. https://doi.org/10.1016/j.knosys.2025.114133

[52] Y. Zhang, Y. Zhang, MixRec: individual and collective mixing empowers data augmentation for recommender systems, in: Proceedings of the ACM on Web Conference 2025, WWW '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 2198–2208. https://doi.org/10.1145/3696410.3714565

[53] V.D. Hoang, Y. Fang, H.W. Lauw, A contrastive framework with user, item and review alignment for recommendation, in: Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, WSDM '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 117–126. https://doi.org/10.1145/3701551.3703530

[54] J. Yu, H. Yin, X. Xia, T. Chen, J. Li, Z. Huang, Self-supervised learning for recommender systems: a survey, IEEE Trans. Knowl. Data Eng. 36 (1) (2024) 335–355. https://doi.org/10.1109/TKDE.2023.3282907