IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 March 2024, accepted 30 March 2024, date of publication 2 April 2024, date of current version 10 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3384376

RESEARCH ARTICLE

Cross-Grained Neural Collaborative Filtering for Recommendation

CHUNTAI LI[1], YUE KOU[1], DERONG SHEN[1], TIEZHENG NIE[1], AND DONG LI[2]
[1]School of Computer Science and Engineering, Northeastern University, Shenyang 110167, China
[2]College of Information, Liaoning University, Shenyang 110036, China

Corresponding author: Yue Kou (kouyue@cse.neu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62072084, Grant 62172082, and Grant 62072086; in part by the Science and Technology Program Major Project of Liaoning Province of China under Grant 2022JH1/10400009; in part by the Natural Science Foundation of Liaoning Province of China under Grant 2022-MS-171; and in part by the Fundamental Research Funds for the Central Universities under Grant N2116008.

ABSTRACT Collaborative Filtering has achieved great success in capturing users' preferences over items. However, existing techniques only consider limited collaborative signals, leading to unsatisfactory results when the user-item interactions are sparse. In this paper, we propose a Cross-grained Neural Collaborative Filtering model (CNCF), which enables recommendation more accurate and explainable. Specifically, we first construct four kinds of interaction graphs to model both fine-grained collaborative signals and coarse-grained collaborative signals, which can better compensate for the high sparsity of user-item interactions. Then we propose a fine-grained collaborative representation learning and design Light Attribute Prediction Networks (*LAPN*) to capture the high-order attribute interactions and enhance the prediction accuracy. Finally, we propose a coarse-grained collaborative representation learning to represent user preferences based on diverse latent intent factors. The experiments demonstrate the high effectiveness of our proposed model.

INDEX TERMS Collaborative filtering, collaborative representation learning, graph neural networks, recommender system.

I. INTRODUCTION

Owing to the rapid development of online services in recent years, more and more commercial websites extensively use recommender systems to enhance their user experiences. Collaborative Filtering (CF) based recommendation provides personalized item suggestions to users according to their historical preferences such as user-item interactions. It generates recommendations based on the assumption that similar users will have common preferences on similar items. Recent studies show that Graph Neural Networks (GNNs) have achieved state-of-the-art performance in modeling high-order relationships for CF [1], [2], [3], [4], [5], [6], [7]. However, most GNNs-based CF models only leverage the original user-item interaction graph to construct the learning task, lacking the explicit exploitation of the auxiliary information in the form of user/item attributes (e.g., age, gender, color, size, or supplier). They usually suffer from the cold-start problem due to severely sparse user-item interactions. Therefore, attribute-aware CF models [8], [9], [10], [11], [12], [13], [14], [15], [16], [17] are proposed to leverage the auxiliary information to enhance recommendations. They aim to provide useful information for more accurate predictions, which is important for many applications such as e-business, entertainment, and social events.

We study the problem of attribute-aware CF for recommendation. Given a user-item pair with their content attributes, we aim to automatically learn the user profile and predict the score of the item for the user. For attribute-aware CF, three issues need to be addressed. First, we need to design a novel model for generating the recommendation that reflects both user preferences and user/item characteristics. Naturally, similar users may generate similar behavioral data. Also, users may be interested in items that are similar to the ones they have historically interacted with. A good model should be able to simultaneously capture as many collaborative signals as possible to compensate for the sparsity of the user-item interactions. Second, we need to

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

design a user/item representation learning that well captures attribute interactions. For example, old people usually like mobile phones with big screens. Hence, in an e-business recommender system, considering the attribute interaction <*age*: *the elderly*, *size*: *big screen*> is more effective than considering the two attributes separately. In addition, different attribute interactions should have different impacts on the final prediction. In the above scenario, the attribute interaction <*age*:*the elderly*, *size*:*big screen*> is more important than <*age*:*the elderly*, *color*:*silver*> for rating prediction. In addition, the higher-order attribute interaction <*age*:*the elderly*, *size*:*big screen*, *price*:*expensive*> is more important than the above two pairwise attribute interactions. Thus considering fine-grained attribute interactions and distinguishing their different impacts to model users/items is a valuable research issue. Finally, we need to represent users based on diverse latent intent factors (e.g., preferences of people of the same age or occupation). For example, influenced by her peers, a young woman may prefer a mobile phone that is light and fashionable. Failing to capture the latent intent factors may lead to a low recommendation quality.

Recently, CF-based recommendation models can be categorized into GNNs-based CF models [1], [2], [3], [4], [5] and attribute-aware CF models [8], [9], [10], [11], [12], [13], [14]. GNNs-based CF models leverage GNNs to model user-item interactions or knowledge graphs, and make a prediction. Attribute-aware CF models take the attribute interactions into account to jointly decide the final predictions. However, they focus on capturing collaborative signals that come either from user-item interactions or attribute interactions. Limited collaborative signals inevitably limit the capability of existing work to leverage the interaction information for producing satisfactory results. Therefore, we propose a Cross-grained Neural Collaborative Filtering model (CNCF), which utilizes the interactions among users, items, and attributes to collaboratively learn embedding vectors for each user and item. Specifically, to alleviate data sparsity and cold-start problems, we consider both attribute interactions and user-item interactions to capture more collaborative signals. Then we collaboratively learn user/item representations by modeling attribute interactions and distinguishing them in a more fine-grained way. Finally, we learn embedding vectors for users based on diverse latent intent factors in a coarse-grained way. We summarise our contributions as follows.

- We construct four kinds of interaction graphs to model both fine-grained collaborative signals and coarse-grained collaborative signals, which can better compensate for the high sparsity of user-item interactions.
- We propose a fine-grained collaborative representation learning, which effectively leverages fine-grained collaborative signals for user/item characteristic learning. We also design Light Attribute Prediction Networks (*LAPN*) to capture the high-order attribute interactions and enhance the prediction accuracy. With this model, the different impacts of attribute interactions can be captured in a fine-grained way to support the model's interpretability better.
- We propose a coarse-grained collaborative representation learning, which utilizes coarse-grained collaborative signals to learn user preferences based on diverse latent intent factors. With this model, the high-order user relationships in the original user-item interaction graph can be captured to improve the recommendation quality.
- We conduct experiments on two real-world datasets to verify the effectiveness of our model. The experimental results demonstrate the high effectiveness of our proposed model for recommendation.

The rest of this paper is organized as follows. Section II reviews the related work. Section III gives an overview of our CNCF model. Section IV describes the major components of CNCF. Section V shows the experimental results and Section VI makes conclusion.

II. RELATED WORK

We review existing literature on two topics closely related to our work, including GNNs-based CF models and attribute-aware CF models.

A. GNNS-BASED CF MODELS

Deep learning has undeniably achieved remarkable success in numerous conventional machine learning domains, including image classification and natural language processing (NLP). Its effectiveness stems from its capacity to discern patterns and features in data. However, much of the data encountered in real-life scenarios deviates from Euclidean space and adheres to non-Euclidean graph structures, such as the inherent structural representations of molecules. To tackle this challenge, researchers have introduced Graph Neural Networks (GNNs) [18], [19], [20], [21], [22], [23] to address data with graph structures. Some graph-based studies [24] use user-item interactions as bipartite graphs to model user interests. However, these approaches typically overlook the user/item attributes, leading to degraded model performance, especially when data is sparse. Some research work treats user attributes differently from item attributes to effectively capture interactions between features [13]. Meanwhile, other studies have proposed automatic selection strategies for feature interactions to improve model performance [10], [25], [26], [27]. Utilizing GNNs-based CF models, several recommender systems have been designed and implemented in the field of social recommendation. These systems employ GNNs to model social relationships and recommend friends or collaborators to users. For instance, [28] introduces the graph conversion capsule link and transforms the social recommendation problem into a graph classification problem for solving. On the other hand, [29] proposes a model-independent social graph denoising module to better learn user social representations. However, these methods either ignore the efficient utilization of fine-grained collaboration signals or ignore the explicit capture of coarse-grained collaboration signals. Our proposed model attempts to capture

both coarse-grained and fine-grained collaboration signals to provide a more comprehensive and efficient solution.

B. ATTRIBUTE-AWARE CF MODELS

Factorization Machine (FM) [8], [30] models each attribute interaction as a dot product of two embedded vectors and aggregates all the modeling results linearly. However, it does not inherently distinguish the varying importance of different interactions. To mitigate this, certain extensions of FM attempt to incorporate the attention mechanism. For example, the attention scores are calculated to differentiate the importance of various feature interactions [9]. In addition to attention mechanisms, other extensions extract nonlinear features by introducing a multilayer perceptron (MLP) into the model [11]. Utilizing attribute-aware CF models, several recommender systems have been designed. For instance, music recommender systems typically take into account attributes such as songwriters, genres, tempos, and other relevant information. Reference [31] employs neural networks to categorize music genres and recommends songs based on three basic attributes: genre, Mel Frequency Cepstrum Coefficients (MFCC) and song tempo [32]. However, these work only focuses on low-order feature interactions. To address this problem, we introduce depth-based and breadth-based messages respectively, to capture the fine-grained high-order collaborative signals.

III. FRAMEWORK OF OUR SOLUTION

In this paper, we propose a novel attribute-aware CF framework, as shown in Figure 1. Given a user/item set with attribute data, user-item interactions, and a target user u, our recommendation predicts the score of u to each item v (v is new to u) and outputs the top-k recommendations with top scores. We construct four kinds of interaction graphs, i.e. attribute-level internal interaction graph (inter-A-A), attribute-level external interaction graph (exter-A-A), social graph among users (U-U) and user-attribute-user hypergraph (U-A-U), to model collaborative signals. It contains three major components, Fine-Grained Collaborative Representation Learning, Coarse-Grained Collaborative Representation Learning, and Light Attribute Prediction Network. 1) Fine-Grained Collaborative Representation Learning captures two kinds of fine-grained collaborative signals, i.e. Attribute-level Internal Collaborative Signals (AICS) and Attribute-level External Collaborative Signals (AECS), to learn user/item representations. 2) Coarse-Grained Collaborative Representation Learning captures two kinds of coarse-grained collaborative signals, i.e. User-level Social Collaborative Signals (USCS) and User-level High-order Collaborative Signals (UHCS), to learn user representations. 3) Light Attribute Prediction Network is used to predict attributes, which is an auxiliary task to make the model improve its understanding of attributes and predict scores better. We will detail these components in Section IV. The important notations used in this paper are listed in Table 1.

TABLE 1. Notations.

Symbols	Definitions and Description
u, v	User u ,Item v
v_i	Attribute i
e_i	The initial embedding of v_i
e_i^{in}	The embedding of v_i via internal interaction
e_i^D	The depth-based embedding of v_i
e_i^B	The breadth-based embedding of v_i
e_i^{ext}	The embedding of v_i via external interaction
e_i^f	The fined-grained embedding of v_i
e_{US}^i	The fine-grained embedding of u_i based on USCS
e_{UH}^i	The fine-grained embedding of u_i based on UHCS
e_{user}^c	The coarse-grained embedding of a user
$\hat{y}$	The predicted preference score

IV. OUR PROPOSED MODEL: CNCF

In this section, we will present our proposed model (CNCF) that predicts the score of how much u likes v.

A. FINE-GRAINED COLLABORATIVE REPRESENTATION LEARNING

First, we introduce how to model two kinds of fine-grained collaborative signals, i.e. AICS and AECS. Then, we discuss how to capture them for representation learning, including AICS-based representation learning and AECS-based representation learning.

1) FINE-GRAINED COLLABORATIVE SIGNALS MODELING

We construct the inter-A-A graphs and the exter-A-A graphs to capture two kinds of fine-grained collaborative signals: AICS and AECS. In our approach, we represent each attribute value (denoted as att) within each attribute domain using a dedicated vector (e), during the embedding stage. The user features are denoted as $U = \left[e_1^u, e_2^u, \cdots, e_{|U|}^u\right]$, the item features are denoted as $I = \left[e_1^i, e_2^i, \cdots, e_{|I|}^i\right]$.

To capture the AICS signals, we construct an inter-A-A for each user/item. First, we categorize attributes into two distinct groups based on their sources: user attributes and item attributes. Then each user/item attribute is treated as a node, and the interaction between two attributes is represented as an edge. In this way, each user/item corresponds to a fully connected subgraph, where the nodes are all the attributes of the user/item and the edges are the interactions between these attributes. Figure 2(a) shows two kinds of inter-A-A, which are used to model the internal attribute interactions for a user and an item respectively.

To capture the AECS signals, we construct an exter-A-A for each user/item. For each user-item pair (u, v), we select one attribute from one part to interact with all attributes of the other part. For example, for each attribute of u, we construct a user-centric exter-A-A to describe its interactions with each attribute of v. The process is repeated for all the attributes of u. We also construct an item-centric exter-A-A for each attribute of v in the same way. Figure 2(b) shows an item-centric exter-A-A and a user-centric exter-A-A respectively.

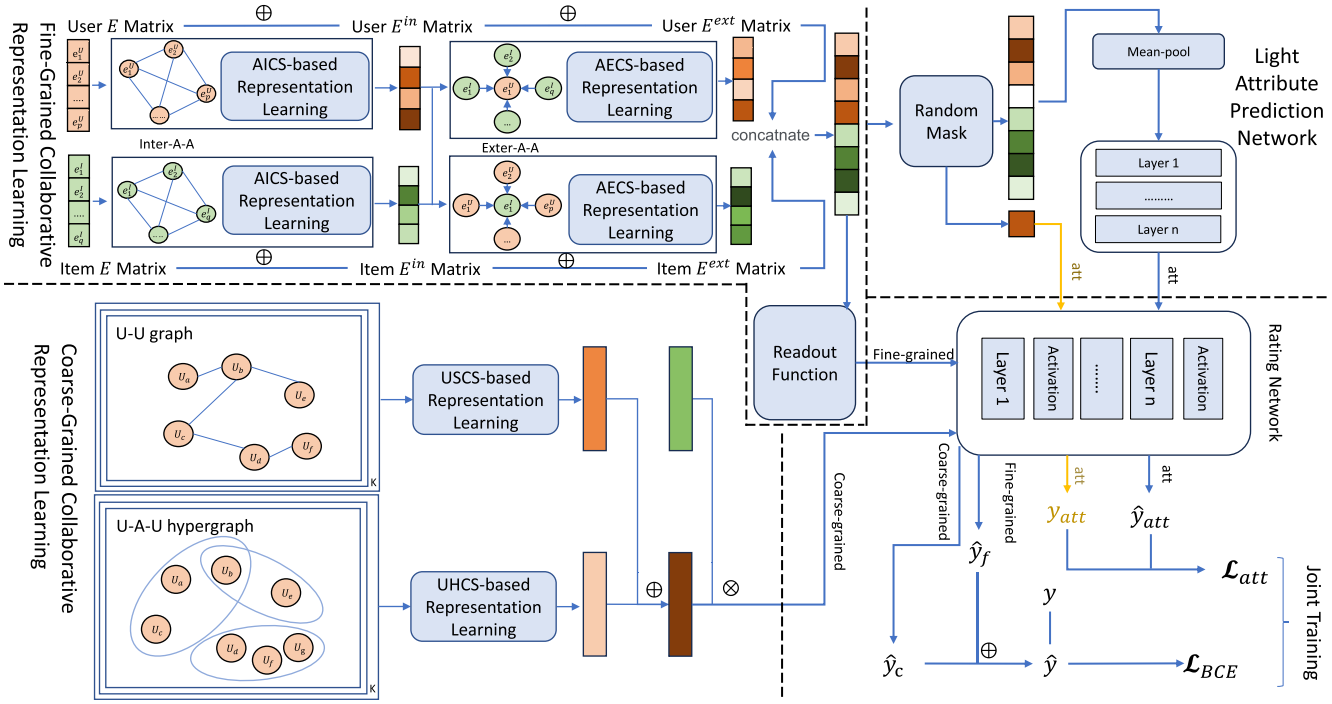
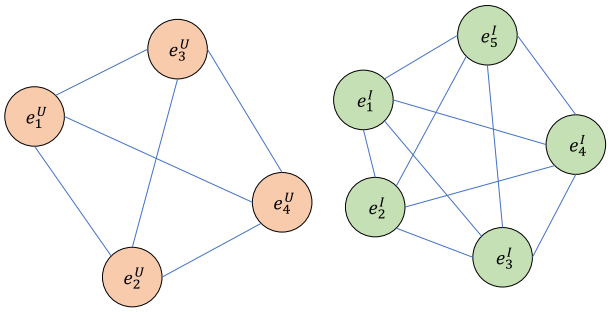
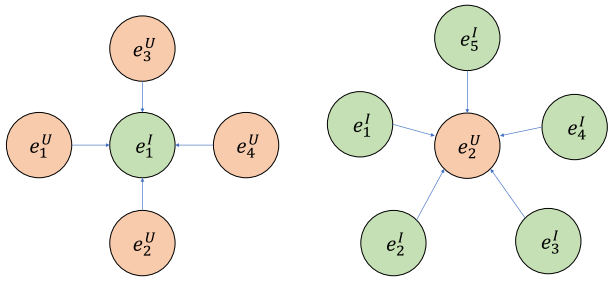


FIGURE 1. Overall framework of CNCF.



(a) Inter-A-A



(b) Exter-A-A

FIGURE 2. Attribute-level internal/external interaction graphs.

node. For inter-A-A, we define a message exchanged between two nodes as the element-wise product of their embeddings:

$$message(v_i, v_j) = e_i \odot e_j \tag{1}$$

where e_i and e_j denote the embeddings of the attribute node v_i and v_j respectively. Here both v_i and v_j are derived from either a user or an item. However, the message defined as (1) and the attribute embeddings do not belong to the same feature space. As a result, the message cannot be directly conveyed to these attribute nodes. Inspired by TransH [33], we employ a transformation matrix to convert the message into the same feature space as the attribute embeddings:

$$message(v_i, v_j) = W_1(e_i \odot e_j) \tag{2}$$

where W_1 denotes the transformation matrix. The transformation matrix is used to convert the message to a specific space. Also, the size of the message can be maintained because a scalar multiplied by a matrix is still a matrix. Then, we aggregate the neighborhood messages to learn attribute representations:

$$e_i^{in} = Aggregate(\{message(v_i, v_j)\}_{j \in N(i)}) \tag{3}$$

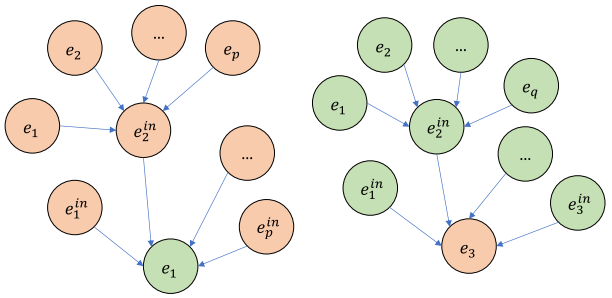
where e^{in} denotes the attribute embedding after internal interaction. In this way, we can capture the AICS.

2) AICS-BASED REPRESENTATION LEARNING

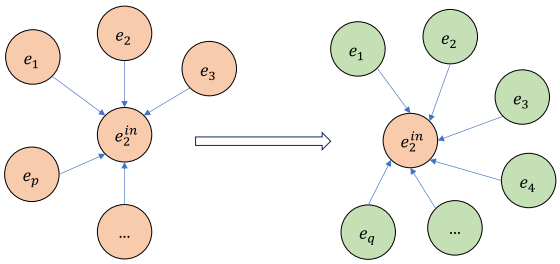
The attribute-level internal interactions are used to capture the AICS signals. Here we employ a Graph Neural Network (GNN) to learn the representation of each user/item attribute

3) AECS-BASED REPRESENTATION LEARNING

For each user-item pair, an external attribute interaction means the interaction between two attributes from the user and the item, respectively. These two attributes are neighbors



(a) Depth-based message passing



(b) Breadth-based message passing

FIGURE 3. Illustration of depth-based/breadth-based message passing.

of each other. To model external attribute interactions, we consider both the depth and the breadth of messages passed between two neighboring attributes.

Suppose v_i and v_j are two attributes that are neighbors of each other. For the depth-based message passed from v_j to v_i, we consider the messages passed between v_j and other attribute nodes that have internal interactions with it. For example, if we recommend a mobile phone to a user, we will consider the impact of the brand of the mobile phone on the user. However, it's also important to consider the interaction between brand and price in this process, as users may have their own opinions about the value of certain brands of phones relative to their price. Therefore, we capture the depth-based message passed from v_j to v_i, which can be represented as:

$$DepthMessage(v_i, v_j) = W_D(e_i \odot e_j^{in}) \tag{4}$$

where W_D denotes the transformation matrix and e_j^{in} denotes the embedding of v_j via internal interactions. The process of depth-based message passing is illustrated in Figure 3(a). Different colors of the nodes indicate that they come from different sources.

Then we aggregate all such depth-based messages passed from v_i's neighbors $N(i)$:

$$e_i^D = Aggregate(\{DepthMessage(v_i, v_j)\}_{j \in N(i)}) \tag{5}$$

where e_i^D denotes the depth-based embedding of v_i via external interactions and $N(i)$ denotes the neighborhood of node v_i.

For the breadth-based message passed from v_j to v_i, we consider the messages passed between v_i and other attribute nodes that have internal interactions with it. For instance, suppose recommend mobile phones to users, we need to consider the internal interaction between users' occupations and their ages. This is crucial because individuals of the same age but with different occupations may exhibit distinct preferences for mobile phones. Therefore, we capture the breadth-based message passed from v_j to v_i, which can be represented as:

$$BreadthMessage(v_i, v_j) = W_B(e_i^{in} \odot e_j) \tag{6}$$

where W_B denotes the transformation matrix. The process of breadth-based message passing is illustrated in Figure 3(b). Then we aggregate all such breadth-based messages passed from v_i's neighbors $N(i)$:

$$e_i^B = Aggregate(\{BreadthMessage(v_i, v_j)\}_{j \in N(i)}) \tag{7}$$

where e_i^B denotes the breadth-based embedding via external interactions.

We incorporate both the depth-based embedding and the breadth-based embedding to obtain the externally interacted embedding of v_i:

$$e_i^{ext} = e_i^D + e_i^B \tag{8}$$

where e_i^{ext} denotes the embedding of v_i via the external interactions. In this way, we can capture the AECS signals.

The final embedding of v_i is defined by aggregating its inherent representation, internal representation, and external representation:

$$e_i^f = e_i + e_i^{in} + e_i^{ext} \tag{9}$$

where e_i denotes the initial embedding of v_i and e_i^f denotes the final embedding of v_i.

B. COARSE-GRAINED COLLABORATIVE REPRESENTATION LEARNING

In the above section, we consider attribute interactions to model fine-grained collaborative signals. In the embedded layer of fine-grained collaborative representation learning, each user/item has an "id" attribute that acts as a unique identifier. We use the embedding of the "id" attribute as the initial coarse-grained embedding of the corresponding user/item. In this section, we consider the interactions between users to model coarse-grained collaborative signals and capture them for user representation learning.

1) COARSE-GRAINED COLLABORATIVE SIGNALS MODELING

First, we construct a social graph among users (U-U) by utilizing the user-item bipartite graph to capture the USCS. Then based on the attributes owned by the user, we build a user-attribute-user hypergraph (U-A-U) to capture the UHCS.

The USCS are used to model the interactions between users. Users should be influenced by people with similar behaviors. To achieve this, we built a U-U graph using the user-item bipartite graph as a basis. We use $U_m = \{i_1, i_2, \cdots, i_{|U_m|}\}$ to denote the items that $user_m$ interacts with, and use A_I to denote the adjacency matrix of the U-U

graph. If two users interact with the same item, there is an edge between them in the U-U graph. Here A_I is calculated as:

$$a_{ij} = \begin{cases} 1, & U_i \cap U_j \neq \emptyset \\ 0, & else \end{cases} \tag{10}$$

The UHCS are used to model the influence of one user on another user due to a common attribute, denoted as att_a. However, many users may share the same attribute. Therefore, we introduce hypergraphs to model the high-order interactions between users. In a typical graph, an edge can connect only two nodes. But in a hypergraph, a hyperedge can connect any number of nodes. We utilize $U_i^A = \left\{att_1, att_2, \cdots, att_{|U_i^A|}\right\}$ to represent the attributes owned by $user_i$. And we denote the set of attributes for all users as $Att = \left\{U_1^A \cup U_2^A \cup \cdots \cup U_m^A\right\}$ and m represents the number of users. Each attribute within Att is considered as a hyperedge. The number of hyperedges is equivalent to the number of attributes in Att.

2) USCS-BASED REPRESENTATION LEARNING

To incorporate users' social information, we utilize graph convolutional neural networks. The embeddings of users are calculated as follows:

$$E^{l+1} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} E^l \Theta^l \tag{11}$$

where D is the diagonal node degree matrix of A, E^l denotes the embedding of the l-th layer of the user, and Θ^l denotes the trainable parameter matrix of the l-th layer. We use E_{US} to denote the output of the last layer and e_{US}^i to denote the embedding of $user_i$.

3) UHCS-BASED REPRESENTATION LEARNING

Just as a typical graph uses an adjacency matrix to represent the relationship between connected nodes, a hypergraph introduces an incidence matrix H to describe the relationship between nodes and hyperedges. Referring to the spectral hypergraph convolution proposed in HGNN [34], the hypergraph convolution of U-A-U is defined as follows:

$$X^{l+1} = \sigma(D^{-1} H W B^{-1} H^T X^l \Theta^l) \tag{12}$$

where D denotes the diagonal vertex degree matrix of the hypergraph U-A-U, W denotes the weight matrix, B denotes the diagonal hyperedge degree matrix, σ denotes the sigmod activation function, X^l denotes the embedding of the l-th layer of the user, and Θ^l is the parameter matrix. Some studies [35] have shown that the activation function is not necessary, so we use the following simplified calculation:

$$E^{l+1} = D^{-1} H B^{-1} H^T E^l \Theta^l \tag{13}$$

We use E_{UH} to denote the output of the last layer and e_{UH}^i to denote the embedding of $user_i$.

Finally, we aggregate both the USCS-based embedding and the UHCS-based embedding to obtain the ultimate user embedding, i.e. the coarse-grained user embedding e_{user}^c.

$$e_{user}^c = e_{US} + e_{UH} \tag{14}$$

C. ENHANCING CNCF WITH LAPN

In practice, it is not feasible to capture the higher-order attribute interactions by stacking network layers, especially when dealing with complex models that are difficult to train. When dealing with limited data, simply increasing the number of layers can cause performance degradation. Therefore, it is essential to enable the model to understand the semantics of attributes and to deeply understand the relationships that exist between these attributes.

There may be a correlation between two attributes that can be a strong basis for a recommendation. For example, if we notice that a person owns an old cell phone with a large font, we can reasonably infer that the person is probably elderly. Therefore, when recommending mobile phones to the elderly, it makes sense to include phones with larger screens as a recommendation result, as there is a correlation between user age and preference for specific mobile phone features. Thus, if the model can understand the correlations between attributes, its performance can be significantly improved. These correlations can provide valuable information and help make accurate decisions.

We hope the model can capture not only the user's preferences for a particular type of item but also the interactions between attributes. These interactions can have both positive and negative impacts. As a positive example, the model can learn about a user's preference for specific item attributes, such as how older individuals tend to favor mobile phones with larger screens. Instead, negative examples cause the model to learn specific attributes of an item that users don't like. To enable the model to better learn these attribute interactions, we design Light Attribute Prediction Networks (*LAPN*). More specifically, for a given user-item pair, we mask out one of their attributes and then use the remaining attributes to predict the value of the masked one:

$$\hat{att}_i = LAPN(\left\{att_j\right\}_{j \in U \cup I \setminus \{i\}}) \tag{15}$$

LAPN can better capture the high-order attribute interactions. When an attribute is masked, the model's ability to predict that attribute depends on the information contained in the remaining attributes. These remaining attributes encapsulate the effect of the masked attribute on them. Therefore, by training *LAPN* to make predictions about each masked attribute, its effect can be learned. During the prediction process, the model distinguishes between different mask attributes based on their effects, which are represented by embeddings. The training process of *LAPN* is as follows: First, an attribute is selected randomly and masked. Second, the embeddings of the remaining attributes are used to predict the embedding of the masked one. Then, for the masked attribute, the scores of its true value and the predicted value are calculated respectively. Finally, the attribute prediction ability of the model is optimized by minimizing the squared

error between them.

$$att_i = f_{fuse}(e_i, e_i^{in}, e_i^{ext}) \tag{16}$$
$$e_{G\setminus i} = Mean - Pooling(\{att_j\}_{j\in U\cup I\setminus\{i\}}) \tag{17}$$
$$\hat{att}_i = APN(e_{G\setminus i}) = W(\sigma(We_{G\setminus i} + b_1)) + b_2 \tag{18}$$
$$\hat{y}_i = Rating(\hat{att}_i)\ \ y_i = Rating(att_i) \tag{19}$$
$$\mathcal{L}_{att} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{20}$$

where f_{fuse} denotes the fusion operation, $e_{G\setminus i} \in R^{1\times d}$ denotes the embedding of the graph with v_i removed, σ denotes the sigmoid function, b_1 denotes the bias, b_2 denotes the bias, $\hat{att}$ denotes the embedding of the predicted attribute, the rating function is implemented via MLP, and N denotes the number of attributes sampled.

D. MODEL TRAINING

The fine-grained preference scores are calculated as follows. For all attributes in each user-item pair, we use a readout function to derive the global attribute graph embedding, which can be expressed as:

$$e^g = Readout(\{e_i^f\}_{i\in U\cup I}) \tag{21}$$

where e^g denotes the global attribute graph embedding. Then, we employ a rating function to predict the user's preference score for the item:

$$\hat{y}_f = Rating(e^g) = \sigma(We^g + b) \tag{22}$$

where σ denotes the sigmoid function, W denotes the weight matrix, and b denotes the bias.

The coarse-grained preference scores are calculated as follows:

$$\hat{y}_c = Rating(e_{user}^c \odot e_i) = \sigma(W(e_{user}^c \odot e_i) + b) \tag{23}$$

where $\odot$ denotes element-wise product and e_i represents the initial embedding of the attribute i. Specifically, e_i is the embedding of the "id" attribute for the item. It is worth noting that we do not aggregate other attributes of the item at this stage. This is because we have already aggregated these attributes during the fine-grained collaborative representation learning process, thus avoiding redundant operations.

Finally, the fine-grained preference score and the coarse-grained preference score are fused to obtain the final preference score $\hat{y}$:

$$\hat{y} = \alpha\hat{y}_f + (1-\alpha)\hat{y}_c \tag{24}$$

where α is used to balance the fine-grained preference score and the coarse-grained preference score.

We adopt the binary cross entropy loss to train the model:

$$\mathcal{L} = -y\log\hat{y} - (1-y)\log(1-\hat{y}) + \lambda_1\mathcal{L}_{att} + \lambda_2(||\theta||_2) \tag{25}$$

where λ_1 is a hyperparameter that balances the recommendation task and the auxiliary task (i.e. the attribute prediction task) and λ_2 is regularization hyperparameter used to prevent model overfitting.

E. DISCUSSION

In this subsection, we discuss the differences between our model and traditional CF-based recommendation models. First, traditional CF-based recommendation models rely solely on capturing collaborative signals from user-item interactions or attribute interactions to improve recommendation accuracy. However, a single type of collaborative signals are relatively limited, and become unavailable especially in the case of sparse data, thus limiting the model's ability to generate satisfactory results for users. In contrast, our model takes into account a wider range of collaborative signals, namely AICS, AECS, USCS, and UHCS, which can effectively mitigate the impact caused by data sparsity. For instance, when a certain type of collaborative signal is unavailable, our model can still rely on the other three types of signals to ensure the quality of recommendation results. Second, traditional CF-based recommendation models primarily concentrate on coarse-grained feature interactions. Nevertheless, fine-grained interactions offer a more profound understanding of user preferences. Consequently, our model employs a fine-grained collaborative learning approach that takes into account intricate attribute interactions and distinguishes their varying impacts on modeling users/items. Finally, traditional CF-based recommendation models solely focus on pairwise attribute interactions, neglecting the significance of high-order interactions, which are crucial for understanding complex user behaviors and preferences. In contrast, our model innovatively utilizes *LAPN* and hypergraphs to capture both high-order attribute interactions and high-order user interactions, respectively. This enhances the model's ability to comprehensively comprehend and model intricate user behaviors and preferences, leading to more accurate and personalized recommendations.

Currently, we evaluate the impact of each attribute on the final rating based on its rating. A higher rating indicates a stronger contribution of the attribute to the final prediction (i.e., it strongly promotes positive interactions between users and items such as liking or purchasing). Conversely, a lower rating suggests a weaker influence. However, this does not imply that attributes with lower ratings are unimportant. Such attributes may facilitate negative interactions between users and items (e.g., disliking or opposing). These attributes are equally valuable in understanding user preferences. Therefore, when predicting the final rating, we do not exclude attributes with lower ratings. To improve computational efficiency by prioritizing the most pertinent and valuable attributes, we will introduce a new methodology in the future to assess the contribution of each attribute and eliminate those "noisy" ones that could potentially hinder the learning process of our model.

V. EXPERIMENT

A. EXPERIMENTAL SETUP

1) DATASETS

We evaluate the performance of CNCF on two real-world datasets: Taobao [5] and MovieLens 1M [36]. Table 2

TABLE 2. Statistics of the datasets.

Dataset	Interaction	User	Item	Att_u	Att_i
Taobao	2,599,463	4,532	371,760	36	434,254
MovieLens 1M	1,149,238	5,950	3,514	30	6,944

provides a summary of these datasets. Taobao consists of click logs from users on the Taobao website. Similar to the MovieLens dataset, we conduct random negative sampling and select users with more than 20 positive ratings for this dataset. MovieLens 1M contains more than 1 million explicit ratings and includes attribute information for both users and items. Ratings exceeding 3 are considered as positive ratings, and an equivalent number of negative samples are randomly selected for each user based on their positive ratings. To ensure data quality, we focus on users with more than 10 positive ratings.

To evaluate the performance, we adopt the metrics commonly used in Top-K recommendation: Precision@K, Normalized Discounted Cumulative Gain (NDCG@K) and Recall@K. They are calculated as follows:

$$Precision@k = \frac{Number_k}{k} \tag{26}$$

$$NDCG@k = \frac{\sum_{i=1}^{k} \frac{2^{rel_i}-1}{log_2(i+1)}}{\sum_{i=1}^{|REL|} \frac{2^{rel_i}-1}{log_2(i+1)}} \tag{27}$$

$$Recall@k = \frac{Number_k}{Number_{all}} \tag{28}$$

where $Number_k$ denotes the number of correctly predicted samples among the samples with the top-k highest scores, $Number_{all}$ denotes the number of positive samples in all samples, $rel_i \in \{0, 1\}$ is 1 when the l-th sample is positive, and REL denotes the optimal ranking of the predicted results, i.e., the positive examples are all ranked at the top. We set K to 10 and 20.[1]

2) IMPLEMENTATION DETAILS

CNCF is implemented with the Pytorch, where the hyper-parameters are configured according to previous research. Adam algorithm is adopted to optimize the model. The dimension of the embedding is 64. The batch size of MovieLens and Taobao is set to 64 and 512, respectively. The value of α is set to 0.9. The learning rate of MovieLens and Taobao is set to 1×10^{-3} and 2×10^{-3}, respectively. The value of λ_1 which controls the weight of the auxiliary task is set to 1. The value of λ_2 for the regularization is set to 1×10^{-5}. We will detail these hyperparameters in Section V-C.

3) BASELINES

To evaluate the performance of our model, we conducted a comparative analysis against six baselines:

- FM [8] employs factorization to capture arbitrary second-order interactions among attributes through dot product.
- AFM [9] calculates attention scores for each interaction result during the aggregation of second-order feature interactions and uses these scores to distinguish the importance of different interactions.
- NFM [11] models pairwise attribute interactions and aggregates interaction results and uses MLP to extract nonlinear features.
- L_0-SIGN [25] models user/item attributes as nodes and attribute interactions as edges. It identifies beneficial interactions, learns node representations through GNN on the generated graph, and aggregates all nodes to obtain the final predicted result.
- GMCF [13] separates user attributes from item attributes and constructs user attribute graphs and item attribute graphs for each data sample. It models interactions within/across attributes, combines interaction results through a gated network, and ultimately predicts results via graph matching.
- HIRS [26] directly generates valuable feature interactions using hypergraphs. The number of generated feature interactions can be customized to be significantly less than all possible interactions.

[1] Our code is available at https://github.com/codeprovided/CNCF.

B. PERFORMANCE COMPARISON

We compare the experimental results of CNCF with the baselines (shown in Table 3). The best performance for each dataset is highlighted in bold, and the best baseline is highlighted by underlined. We note the following key observations:

First, AFM and NFM outperform FM due to their utilization of the attention mechanism and neural networks, respectively. However, they overlook the high-order interactions among attributes, whereas our model, CNCF, takes these interactions into account, resulting in superior performance compared to AFM and NFM.

Second, our model outperforms L_0-SIGN, GMCF, and HIRS primarily for two reasons. On one hand, these baselines solely consider fine-grained collaborative signals while neglecting coarse-grained ones, and L_0-SIGN and HIRS fail to account for attribute sources. On the other hand, L_0-SIGN and GMCF do not consider high-order interactions among attributes, which our model effectively captures.

Third, compared with the baselines, our CNCF achieves better performance on both datasets. On Taobao, CNCF outperforms other baselines by approximately 2.15%, 1.67%, and 3.73% in precision@20, recall@20, and NDCG@20, respectively. Similarly, on MovieLens 1M, CNCF outperforms benchmark models by about 0.8%, 0.5%, and 0.8% in precision@20, recall@20, and NDCG@20, respectively. This suggests that CNCF can effectively model attribute interactions by capturing both fine-grained and coarse-grained collaborative signals.

Fourth, compared with MovieLens 1M, the performance improvement on Taobao is more obvious. That's because Taobao shows stronger data sparsity than MovieLens 1M. Our model explicitly encodes two different granularity of

TABLE 3. Performance comparison of our model with baselines.

Method	Taobao(%)						MovieLens 1M(%)					
	P@10	P@20	R@10	R@20	N@10	N@20	P@10	P@20	R@10	R@20	N@10	N@20
FM	8.784	8.344	16.145	30.076	12.889	18.944	73.331	56.088	93.806	95.205	93.105	93.461
AFM	9.029	8.476	16.490	30.408	13.177	19.213	73.555	56.206	94.052	95.322	93.304	93.604
NFM	9.146	8.557	16.670	30.756	13.416	19.521	73.707	56.377	94.204	95.513	93.465	93.799
L_0-SIGN	9.248	8.540	16.772	30.868	13.607	19.481	74.065	56.507	94.598	95.666	93.937	94.171
HIRS	9.359	8.621	17.039	30.985	13.645	19.542	74.151	56.519	94.674	95.673	94.027	94.186
GMCF	9.274	8.543	16.915	30.651	13.521	19.477	74.221	56.617	94.758	95.778	94.181	94.374
CNCF	**9.640**	**8.806**	**17.600**	**31.502**	**14.226**	**20.272**	**74.988**	**57.071**	**95.540**	**96.260**	**95.077**	**95.129**
$\Delta\%$	3	2.15	3.29	1.67	4.25	3.73	1.03	0.8	0.83	0.5	0.95	0.8

collaborative signals, allowing it to capture more comprehensive information and better address data sparsity.

C. PARAMETER SENSITIVITY

We evaluate the impact of the hyper-parameters on MovieLens 1M (shown in Figure 4).

1) EFFECT OF THE BATCH SIZE

The setting of batch size has an intricate relationship with weight updating and significantly affects the generalization performance of the model. When the batch size is too small or too large, it may not accurately determine the descent direction, potentially resulting in training oscillations. Therefore, both too small and too large batches can affect the performance of the model, which emphasizes the importance of choosing the appropriate batch size. Figure 4 (a) depicts the performance of our model w.r.t. different batch sizes. We observe that when the batch size is 64, the performance becomes the best. Hence, we set the batch size to 64 for MovieLens 1M.

2) EFFECT OF THE EMBEDDING SIZE

The size of the embedding dimension has a significant effect on the expressiveness of the model. As shown in Figure 4 (b), when the embedding dimension is too small, the model lacks the necessary expressiveness. On the contrary, when the embedding dimension is too large, the model is prone to overfitting. Therefore, choosing the appropriate embedding dimension is the key to affecting the performance of the model. We set the embedding size to 64.

3) EFFECT OF THE VALUE OF λ_1

The parameter λ_1 plays a crucial role in controlling the weight of self-supervised task learning. If λ_1 is set too small, the auxiliary task may not be effectively learned. Instead, the model may over-prioritize learning for the auxiliary task at the expense of the recommendation task. Hence, it is essential to find an appropriate value for it to balance the recommendation task and the auxiliary task. Figure 4 (c) shows the performance of our model w.r.t. different λ_1. We observe that when λ_1 is 1, the performance becomes the best.

TABLE 4. Importance of each component in CNCF.

Method	Taobao (%)					
	P@10	P@20	R@10	R@20	N@10	N@20
w/o pre	9.612	8.797	17.518	31.494	14.185	20.263
w/o coarse+pre	9.609	8.790	17.557	31.422	14.205	20.246
w/o fine+pre	7.564	7.428	13.705	26.751	11.272	16.906
w/o coarse	9.612	8.801	17.535	31.499	14.193	20.269
CNCF	**9.640**	**8.806**	**17.600**	**31.502**	**14.226**	**20.272**
Method	MovieLens 1M (%)					
	P@10	P@20	R@10	R@20	N@10	N@20
w/o pre	74.786	56.914	95.336	96.092	94.845	94.911
w/o coarse+pre	74.700	56.920	95.243	96.101	94.769	94.899
w/o fine+pre	71.301	54.522	91.747	93.568	91.013	91.515
w/o coarse	74.911	57.054	96.136	93.568	94.984	95.074
CNCF	**74.988**	**57.071**	**95.540**	**96.260**	**95.077**	**95.129**

4) EFFECT OF THE NUMBER OF SAMPLES

The number of samples refers to the number of predictive attributes we sample for each user-item pair. Too few samples hinder the model's ability to effectively learn attribute interactions. Conversely, too many samples can cause the model to overfit the data. Figure 4 (d) shows the performance of our model w.r.t. different number of samples. When the number of samples is 80, the performance becomes the best.

D. ABLATION ANALYSIS

To study the effectiveness of each component in the proposed model, we design the following variants:

- w/o pre: Remove the auxiliary task of attribute prediction.
- w/o coarse+pre: Remove coarse-grained representation learning and the auxiliary task of attribute prediction.
- w/o fine+pre: Remove fine-grained representation learning and the auxiliary task of attribute prediction.
- w/o coarse: Remove coarse-grained representation learning.

The results are shown in Table 4. We note the following key observations: First, removing the auxiliary task of attribute prediction leads to the degradation of model performance, indicating that the attribute prediction task plays a crucial role in enhancing the model's understanding of attribute interactions, and helps to better model user interests. Second, eliminating coarse-grained representation learning and the auxiliary task while retaining fine-grained representation learning results in model performance degradation. This highlights the significance of coarse-grained representation

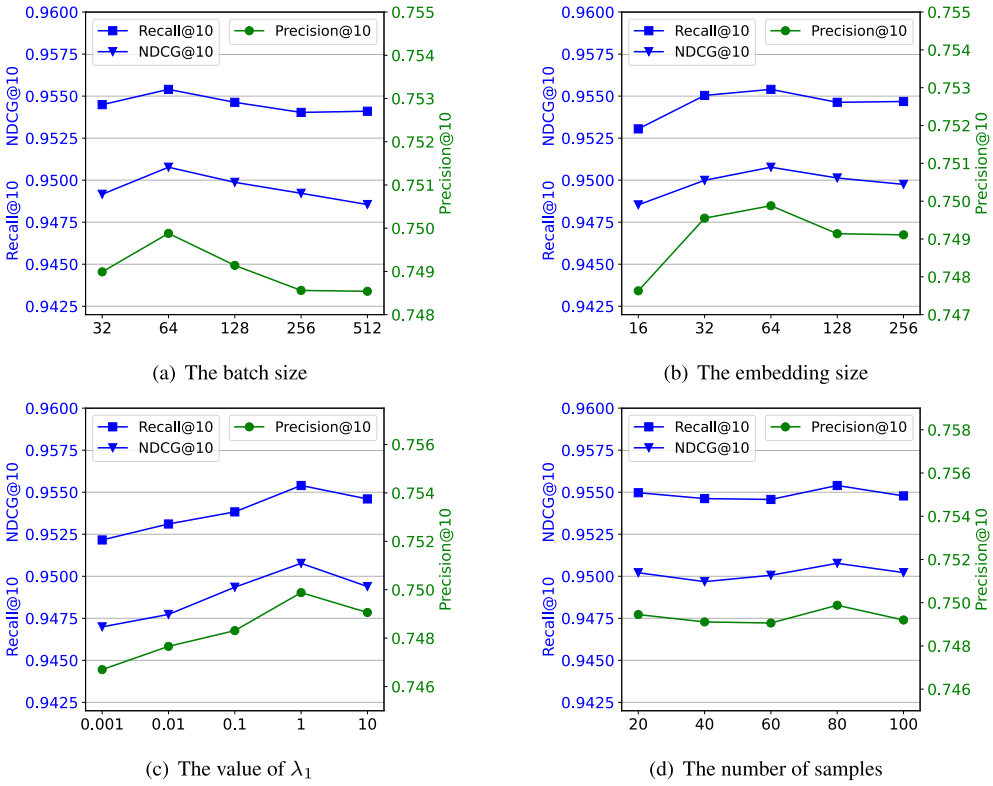


FIGURE 4. Effects of the model parameters on MovieLens 1M.

learning, as it can capture information that fine-grained representation learning alone may overlook. Third, the removal of fine-grained representation learning and the auxiliary task, while maintaining only coarse-grained representation learning, leads to a substantial decrease in model performance. This observation emphasizes the crucial role that fine-grained representation learning plays in the recommendation process. Finally, when coarse-grained representation learning is removed and only fine-grained representation learning and the auxiliary task are retained, the auxiliary task plays a beneficial role in enhancing the effectiveness of fine-grained representation learning.

E. CASE STUDY

In this section, we conduct case studies to evaluate whether the attributes show semantic meaning that provides potential explanations of the predictions. Specifically, we use the learned attribute embeddings from the MovieLens 1M dataset. We calculate the strength of attribute interactions (between user attributes and item attributes) based on attribute embeddings. We compare the strength of attribute interactions based on the user-centric exter-A-A and the item-centric exter-A-A respectively. The former is to use the users' attribute nodes as the target nodes during the messaging phase. While the latter is to use the items' attribute nodes as the target nodes. The darker the color, the higher the strength of attribute interactions. There are 21 different types

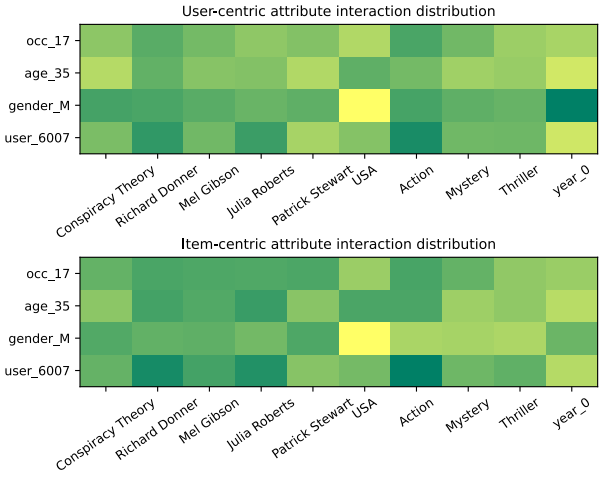


FIGURE 5. The case studies on MovieLens 1M.

of occupations in MovieLens and *occ*_17 means the 17th occupation, i.e. technician/engineer. Similarly, *year*_0 means the year 1997.

As shown in Figure 5, the user-centric attribute interaction distribution is essentially similar to the item-centric attribute interaction distribution. We observe that users with different attributes have different preferences for items. These observations show our model's ability to provide potential explanations about the prediction results at the

attribute level. For example, we recommend the movie (*name*: *Conspiracy Theory*) to the user (*id*: *user*_6007) because the three pairs of attributes (i.e. $< id : user\_6007, genre : Action >$, $< id : user\_6007, director : Richard\ Donner >$ and $< id : user\_6007, Actress : Julia\ Roberts >$) all have strong attribute interactions. The reason for the recommendation is that the user is likely to prefer action movies directed by Richard Donner and starring Julia Roberts.

VI. CONCLUSION

In this paper, we propose a Cross-grained Neural Collaborative Filtering model (CNCF), which enables recommendation more accurate and explainable. Specifically, we construct four kinds of interaction graphs to model both fine-grained collaborative signals and coarse-grained collaborative signals, which can better compensate for the high sparsity of user-item interactions. We propose a fine-grained collaborative representation learning and capture the high-order attribute interactions to enhance prediction accuracy. We propose a coarse-grained collaborative representation learning to learn user preferences based on diverse latent intent factors. Extensive experiments demonstrate the effectiveness of our model. For future work, we will consider the dynamics of collaborative signals and incorporate the review information into recommendation to enhance interpretability.

REFERENCES

[1] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 639–648.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[3] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2019, pp. 165–174.

[4] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep interest evolution network for click-through rate prediction," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, pp. 5941–5948.

[5] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proc. SIGKDD*, 2018, pp. 1059–1068.

[6] J. Wu, X. He, X. Wang, Q. Wang, W. Chen, J. Lian, and X. Xie, "Graph convolution machine for context-aware recommender system," *Frontiers Comput. Sci.*, vol. 16, no. 6, Dec. 2022, Art. no. 166614.

[7] J. Luo, M. He, W. Pan, and Z. Ming, "BGNN: Behavior-aware graph neural network for heterogeneous session-based recommendation," *Frontiers Comput. Sci.*, vol. 17, no. 5, Oct. 2023, Art. no. 175336.

[8] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2010, pp. 995–1000.

[9] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," in *Proc. IJCAI*, 2017, pp. 3119–3125.

[10] B. Liu, C. Zhu, G. Li, W. Zhang, J. Lai, R. Tang, X. He, Z. Li, and Y. Yu, "AutoFIS: Automatic feature interaction selection in factorization models for click-through rate prediction," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2020, pp. 2636–2645.

[11] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Aug. 2017, pp. 355–364.

[12] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 1725–1731.

[13] Y. Su, R. Zhang, S. M. Erfani, and J. Gan, "Neural graph matching based collaborative filtering," in *Proc. SIGIR*, 2021, pp. 849–858.

[14] F. Huang, Z. Wang, X. Huang, Y. Qian, Z. Li, and H. Chen, "Aligning distillation for cold-start item recommendation," in *Proc. 46th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2023, pp. 1147–1157.

[15] M. Blondel, M. Ishihata, A. Fujino, and N. Ueda, "Polynomial networks and factorization machines: New insights and efficient training algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 850–858.

[16] Z.-X. Zhan, M.-K. He, W.-K. Pan, and Z. Ming, "TransRec++: Translation-based sequential recommendation with heterogeneous feedback," *Frontiers Comput. Sci.*, vol. 16, no. 2, Apr. 2022, Art. no. 162615.

[17] E. Xu, Z. Yu, N. Li, H. Cui, L. Yao, and B. Guo, "Quantifying predictability of sequential recommendation via logical constraints," *Frontiers Comput. Sci.*, vol. 17, no. 5, Oct. 2023, Art. no. 175612.

[18] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2008.

[19] Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang, "Fi-GNN: Modeling feature interactions via graph neural networks for CTR prediction," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manag.*, Nov. 2019, pp. 539–548.

[20] S. Tang, K. Yao, J. Liang, Z. Wang, and J. Liang, "Graph neural networks with interlayer feature representation for image super-resolution," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, 2023, pp. 652–660.

[21] B. Li, T. Guo, X. Zhu, Q. Li, Y. Wang, and F. Chen, "SGCCL: Siamese graph contrastive consensus learning for personalized recommendation," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, Feb. 2023, pp. 589–597.

[22] M. Chen, C. Huang, L. Xia, W. Wei, Y. Xu, and R. Luo, "Heterogeneous graph contrastive learning for recommendation," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, Feb. 2023, pp. 544–552.

[23] J. Zhao, Q. Wen, M. Ju, C. Zhang, and Y. Ye, "Self-supervised graph structure refinement for graph neural networks," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, 2023, pp. 159–167.

[24] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[25] Y. Su, R. Zhang, S. Erfani, and Z. Xu, "Detecting beneficial feature interactions for recommender systems," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 5, pp. 4357–4365.

[26] Y. Su, Y. Zhao, S. Erfani, J. Gan, and R. Zhang, "Detecting arbitrary order beneficial feature interactions for recommender systems," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2022, pp. 1676–1686.

[27] W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang, "Autoint: Automatic feature interaction learning via self-attentive neural networks," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manag.*, 2019, pp. 1161–1170.

[28] X. Liu, X. Li, G. Fiumara, and P. De Meo, "Link prediction approach combined graph neural network with capsule network," *Exp. Syst. Appl.*, vol. 212, Feb. 2023, Art. no. 118737.

[29] W. Ma, Y. Wang, Y. Zhu, Z. Wang, M. Jing, X. Zhao, J. Yu, and F. Tang, "MADM: A model-agnostic denoising module for graph-based social recommendation," in *Proc. 17th ACM Int. Conf. Web Search Data Mining*, 2024, pp. 501–509.

[30] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[31] D. Kostrzewa, W. Mazur, and R. Brzeski, "Wide ensembles of neural networks in music genre classification," in *Proc. Int. Conf. Comput. Sci.*, 2022, pp. 64–71.

[32] D. Kostrzewa, J. Chrobak, and R. Brzeski, "Attributes relevance in content-based music recommendation system," *Appl. Sci.*, vol. 14, no. 2, p. 855, Jan. 2024.

[33] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. AAAI Conf. Artif. Intell.*, 2014, vol. 28, no. 1, pp. 1–8.

[34] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 3558–3565.

[35] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6861–6871.
[36] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Jan. 2016.

CHUNTAI LI received the bachelor's degree from Northeast Forestry University, China. He is currently pursuing the M.S. degree with the School of Computer Science and Engineering, Northeastern University, Shenyang, China. His research interests include recommender systems and deep learning.

YUE KOU received the Ph.D. degree in computer software and theory from Northeastern University, Shenyang, China, in 2009. She is currently an Associate Professor with the School of Computer Science and Engineering, Northeastern University. Her research interests include social network analysis and mining, recommender systems, and web data management.

DERONG SHEN received the Ph.D. degree in computer software and theory from Northeastern University, Shenyang, China, in 2004. She is currently a Professor with the School of Computer Science and Engineering, Northeastern University. Her research interests include data integration, big data processing, and web data management.

TIEZHENG NIE received the Ph.D. degree in computer software and theory from Northeastern University, Shenyang, China, in 2009. He is currently an Associate Professor with the School of Computer Science and Engineering, Northeastern University. His research interests include big data management and data integration.

DONG LI received the Ph.D. degree in computer software and theory from Northeastern University, China, in 2019. He is currently an Associate Professor with the School of Information, Liaoning University, Shenyang, China. His research interests include social networks analysis and data mining.